Learning to pronounce written words : a study in inductive language learning

Learning to pronounce written words

A study in inductive language learning

Antal P. J. van den Bosch

Voor Anne-Marie en . . .

Learning to pronounce written words

A study in inductive language learning

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan
de Universiteit Maastricht,
op gezag van de Rector Magnificus, Prof. mr M. J. Cohen,
volgens het besluit van het College van Decanen,
in het openbaar te verdedigen op
donderdag 11 december 1997 om 16.00 uur

door

Antal P. J. van den Bosch

Promotor: Prof. dr H. J. van den Herik

Leden van de beoordelingscommissie:
Prof. dr P. T. W. Hudson (voorzitter)
Dr D. W. Aha (NRL Navy Center for Applied Research in AI)
Dr W. M. P. Daelemans (Katholieke Universiteit Brabant)
Prof. dr M. Papazoglou (Katholieke Universiteit Brabant)
Prof. dr H. Visser

ISBN 90-801577-2-4
Uitgeverij Phidippides, Cadier en Keer

Cover design: the author using POVRay 3.1

Contents

Preface

The investigations described in this thesis and the writing of it have given me much pleasure. I now take equal pleasure in acknowledging all those who have helped me along the way. The work in this thesis would have been impossible without them.

I would like to start by thanking my coaches at the Universiteit Maastricht. First, Jaap van den Herik, my supervisor, with his catching enthusiasm. He has guided me as a true teacher along the paths of how to become a scientist. I shall not easily forget his lessons, notably those on scientific writing, which I have tried to put into practice – any mistakes remaining are my own. I am very grateful to Jaap for taking care of the enjoyable, well-equipped environment of the Department of Computer Science.

Ton Weijters and Eric Postma have been a continuous source of ideas, accurate criticism, sympathy and good humour. I wish to express my gratitude for their support and sincerity, and for letting me tap their expertise. They have urged me to make implicit thoughts explicit, so that these thoughts could either be dismissed as nonsense or be worked out. Their involvement has made the project work.

Besides my three Maastricht advisors, I want to thank Walter Daelemans for his ideas and the scientific background he introduced me to in my student years. Walter's work on machine learning of natural language provided a major part of the scientific basis of this thesis. For many years he has stimulated me by his hunches, hypotheses, and love for empirical work.

The work described in this thesis has also benefited from ideas, suggestions, and concrete contributions from many other people. I want to thank David Aha for his suggestions and discussions, for pointing out new and relevant issues in machine learning. Maria Wolters has been very helpful and patient in reading draft versions of the thesis; I especially appreciate her suggestions on the linguistic aspects. The members of the Tilburg ILK group

have provided excellent suggestions on methodology and algorithms – in particular, I wish to thank Jakub Zavřel.

Jo Beerens and his predecessor Maarten van der Meulen have provided and maintained excellent computer facilities at the Department of Computer Science in Maastricht. Half but not quite in jest, I wish to thank the computers for having repeatedly learned the pronunciation of English words. Even after hundreds of times learning it all over again, they hardly complained.

Although computers may be a powerful factor in this research, the human factor is overpowering. The people at the Department of Computer Science of the Universiteit Maastricht have made my stay a pleasant one. Fred Wan has been more than just the fellow inhabitant of the attic – his sharp mind and friendship are much appreciated, as are Ruud van der Pol's serious and comic contemplations of science and life. Joke Hellemons and Sabine Vanhouwe have done a great job in keeping administrative matters organised.

I would like to thank the Netherlands Organisation for Scientific Research (NWO) for their financial travelling support, and the Centre for Lexical Information in Nijmegen for making available their lexical corpora.

My friends Jeanôt, Tim, René, and Peter have successfully diverted my attention at many occasions, and so has Kees Razenberg – I am looking forward to continuing our projects.

Finally, I would like to thank my family who have never stopped giving me their love and trust. I consider myself very lucky to be the son of Jan and Addie and the brother of Paul van den Bosch. Most important of all has been the love and companionship of my wife Anne-Marie. During the final part of my thesis work, she had to divide her incredible support over me and a third, which has given a whole new sound to the word **expecting**.

Chapter 1

Introduction

1.1 Learning word pronunciation

Learning to pronounce written words means learning the intricate relations between a language's writing system and its speech sounds. When children learn to read and write in primary school they face such a learning task, as do students when mastering the writing system, the speech sounds, and the vocabulary of a language different from their mother tongue. Learning to pronounce words can also be modelled on computers. The latter, rather than simulating learning to pronounce written words in humans, is the topic of the present study. In contrast with humans, machines can be modelled (i.e., realised, set up) in such specific ways that the pronunciation of written words is modelled on these machines. For instance, a machine can be set up to accommodate a data base of representations of word-pronunciation knowledge, without having learned any of those representations by itself: it is hardwired in memory by the system's designer. In fact, the hardwiring of word-pronunciation knowledge is common practice in the development of speech synthesizers (Allen, Hunnicutt, and Klatt 1987; Daelemans 1987).

A major part of language-engineering work on word-pronunciation applications has been based on mainstream linguistic theories which consider only the modelling of word-pronunciation knowledge to be of scientific interest. The American linguist Noam Chomsky can be seen as the principal promoter of this tradition. His work on syntax (Chomsky, 1957), and later work on phonology (Chomsky and Halle, 1968) has influenced linguistics deeply and across the board from the 1950s onwards (Chomsky, 1957; Piatelli–Palmarini, 1980).

Despite the influential arguments of Chomskyan linguistics against the existence of a generic learning method capable of language learning, the possibility of the existence of such a method has been conjectured and investigated within the area of *linguistic structuralism* (Robins, 1997). The field of linguistic structuralism has appeared and reappeared under the names of descriptive, quantitative, statistical, or corpus-based linguistics from the 1930s onwards (Robins, 1997).

Thus, two contrasting views exist on the learnability of word pronunciations by a generic learning method: the Chomskyan view on the one hand, and the linguistic-structuralist view on the other hand. To gain a better understanding of the gap between the two views to language learning, Section 1.2 introduces *inductive language learning* as our interpretation of the linguistic-structuralist view, and sketches the historical line of research in linguistics both in favour of and against a generic method for language learning.

1.2 Inductive language learning

We put forward *inductive language learning* as a generic method for language learning to be employed in our study. The fundamental idea behind inductive language learning is that a major part of language can be modelled by the *analogy principle* (De Saussure, 1916; Glushko, 1979; Yvon, 1996; Lepage and Shin-ichi, 1996; Daelemans, 1996b) stating that *similarities between sequences of one or more symbols of language tend to hold for corresponding sequences of symbols at different levels of language.* A level is a representation domain, in which language is represented as variable-length sequences of symbols. For example, on the level of writing, the letters of the alphabet constitute the symbols, and words or sentences are represented as sequences of letters. On the phonemic speech level, the different phonemes of the language are the symbols, and words or utterances are represented as sequences of phonemes. We will return to describing the different levels discerned generally in word pronunciation in Chapter 2; cf. Figure 2.10 for an illustration.

The analogy principle integrates two types of relations between sequences of one or more language symbols: (i) *similarities* between sequences of language symbols at the same level, and (ii) *correspondences* of sequences of symbols at one level with sequences of symbols at another level. To define similarity and correspondence, we provide simple examples from the domain which is the object of our study, English word pronunciation. We start by identifying two levels: writing and speech. In alphabetic (or logo-

graphic, Coulmas, 1989) writing systems, such as that of English, words are represented on the level of writing by sequences of letters which make up the word's *spelling*: for example, the words **bed** and **bid** are each composed of a sequence of three letters. On the level of speech, words are represented by sequences of speech sounds which make up the word's *pronunciation*: for example, the words **bed** and **bid** are represented on the level of speech as the three-phoneme sequences /bɛd/ and /bɪd/, respectively. Similarities exist between the spellings **bed** and **bid**; restricting ourselves in this example to matching letters at identical word positions, **bed** and **bid** are similar in that they share two letters: a word-initial **b** and a word-final **d**. The pronunciations, or *phonemic transcriptions* /bɛd/ and /bɪd/ share two phonemes (the commonly-accepted symbols used to denote speech sounds): a word-initial phoneme /b/ and a word-final phoneme /d/.

Correspondences exist between **bed** and its phonemic transcription /bɛd/, and between **bid** and its phonemic transcription /bɪd/. Correspondences express the relations between written words and their pronunciations, and are given by the (potentially arbitrary) pronunciation employed by language users and stored in pronunciation lexicons.

Thus, the similarly-spelled words **bed** and **bid** have the similar corresponding pronunciations /bɛd/ and /bɪd/, respectively. Consider, for example, also the pairs **book** and **books**, and their pronunciations /buk/ and /buks/; **believe** and **relieve**, and their pronunciations /bəliv/ and /rəliv/; **analytically** and **anarchistically** and their pronunciations /ænəlɪtɪkəlɪ/ and /ænəkɪstɪkəlɪ/. In all examples, identical letter groups within pairs of words are mirrored by identical phoneme groups in the corresponding pronunciations. Matching identical letters or phonemes at identical positions is a concededly simplistic definition of similarity; consider, for example, the words pair **bid** and **abide**, or **buck** and **buy**. English word pronunciation is infested with these types of inconsistencies; dealing with them in inductive language learning means that the similarity function should be able to exploit contextual information, even if this means including the whole word in the context. While the **bed**/**bid** example used here refers to similarity as counting single letters at identical word positions, computing similarity should be essentially unbiased towards the number of letters and phonemes between which correspondences may be learned. We will return to the issue of computing similarities between (parts of) words in Chapter 2.

Figure 1.1 visualises the similarity and correspondence relations as assumed by the analogy principle for the general case of word pronunciation.

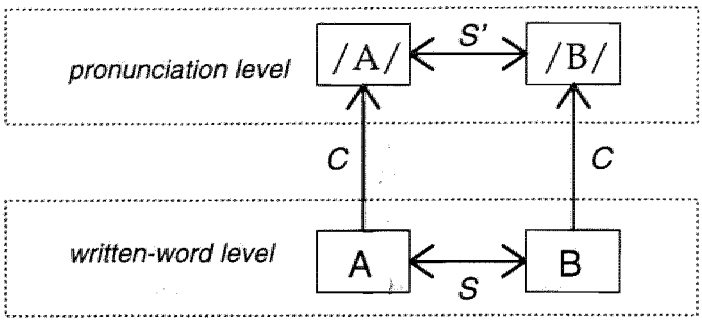


Figure 1.1: Visualisation of the analogy principle in word pronunciation. Word **A** is as similar to word **B** as the pronunciation of **A**, /A/, is to the pronunciation of **B**, /B/. Correspondence relations are denoted by C; S and S' denote the similarity relations existing between words and between pronunciations, respectively.

Given a word **A** and its corresponding pronunciation /A/, and a word **B** and its corresponding pronunciation /B/, if **A** and **B** stand in a similarity relation, then /A/ and /B/ stand in a similarity relation.

In any alphabetic languages, the similarity relations among words (indicated by S in Figure 1.1) and among pronunciations (indicated by S' in Figure 1.1) often do not yield the same quantifications of similarity. For the example word-pronunciation pairs **bed** – /bɛd/ and **bid** – /bɪd/, both S and S' yield similarities of two matching elements at identical word positions, but other examples of word pairs (e.g., **though** – /ðəʊ/ and **through** – /θru /, **hear** – /hɪə/ and **heart** – /hɑɪt/) readily show that S may yield a higher similarity between spelling strings than the similarity between the corresponding pronunciations yielded by S'. The opposite also occurs in English, e.g., **little** and **victual** have a rather different spelling, but their corresponding pronunciations are highly similar: /lɪt$_{\text{ˌ}}$l / and /vɪt$_{\text{ˌ}}$l /, respectively. This lack of complete equivalence of S and S' is due to the fact that letters in alphabetic writing systems essentially originate from denoting speech sounds in a one-to-one manner, but have shifted gradually or arbitrarily towards many-to-one, one-to-many, and many-to-many relations (Röhr, 1994). Yet, the original one-to-one relations still shine through; they do occur often and allow the analogy principle to be readily applicable for a considerable number of cases. Thus, for languages with alphabetic writing systems, we assume that $S \approx S'$, where '$\approx$' is used rather than '$=$' to denotes the distortion of the original one-to-

one mappings between spelling and speech. It follows that for pictographic and ideographic writing systems, such as that of Chinese, $S \neq S'$: in these writing systems, similarities between written words do not hold between the pronunciations of those words.

While the analogy principle describes a static phenomenon, it can be employed actively in pronouncing words. The problem for a word-pronunciation system is that it cannot simply reproduce some known word pronunciation when presented with a word never seen before; it is essential for the system to be able to produce a best guess for the new word's pronunciation. It would be advantageous to be able to extract knowledge from the word-pronunciation correspondences encountered earlier and stored in memory. To this end *inductive language learning*[1] can be employed. Inductive language learning is the combination of the analogy principle with the general reasoning method *induction*. A generic definition of induction adopted here is *a method of reasoning on the basis of known facts to produce general rules or principles*. Flach (1995) remarks in the introductory chapter of his thesis that while the above-mentioned definition appears to work, there is no well-established definition of induction. Cf. Flach (1995) and Harnad (1982) for further discussions on the definition of induction from a logical and cognitive perspective, respectively.

Figure 1.2 illustrates the process of inductive language learning specific for the word-pronunciation task, in its most basic and general form. It is an adaptation of Figure 1.1, in which the pronunciation of word **B** is assumed unknown (denoted by the question mark). For example, supposing that **A** is **bed**, and that **B** is **bid**, the analogy principle states that the unknown pronunciation of **bid** is as similar to /bɛd/ as **bid** is to **bed**. The two words are indeed similar, since they share the first and last letters. As said, sharing letters at identical word positions is a weak measure of similarity, but it suffices in this example.

Induction is reasoning on the basis of known facts to produce general rules or principles. The first fact that comes to mind is that **A** corresponds to /A/, but that is of no use in finding the unknown pronunciation of **B** since **B** is not equal to **A**. Since the analogy principle assumes correspondences between sequences of letters and phonemes, facts should be sought at the level of letters and phonemes; for example, that **b** corresponds with /b/, that

[1] Daelemans (1995) uses the alternative summarising term *similarity-based learning of language tasks*. Similarity-based learning is often used as a synonym for inductive learning (Shavlik and Dietterich, 1990, p. 1).

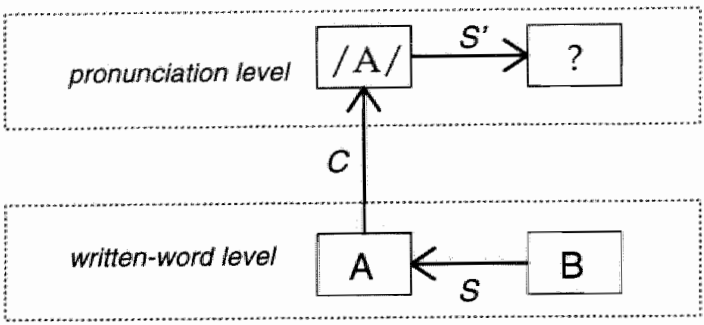


Figure 1.2: Visualisation of the analogy principle used in combination with induction. The unknown pronunciation of **B** is induced from that of **A** insofar as **A** is similar to **B**.

e corresponds with /ɛ/, and that **d** corresponds with /d/. These facts can be established automatically from known correspondences, e.g., in this example, by assuming letters and phonemes at identical positions to be corresponding. Establishing these facts and storing them in memory (be it long-term memory or working memory) for later use is done in two steps.

The first step in inductive learning. This first step may occur in a separate first phase in which all correspondences between letters and phonemes are established on the basis of known word-pronunciation correspondences, after which this stored knowledge can be used to process new words, while it can also occur *on demand*: establishing letter-phoneme correspondences on the basis of known word-pronunciation correspondences only when they are needed, for example, when processing an unknown word.

The second step of inductive learning is the induction of pronunciations given some new input, on the basis of information searched in the first step. Having stored in the first step that **b** corresponds with /b/, and **d** with /d/, it can be induced from these facts that the pronunciation of **bid** starts with /b/ and ends with /d/. To induce the phonemic correspondence of the middle letter **i**, it is necessary to have another correspondence in memory, of a word **D** (e.g., **lid**) and its corresponding pronunciation (/lɪd/) with a middle **i**. This would be a minimal solution; another would be to match a new word with an unknown pronunciation against all known word-pronunciation correspondences ever encountered and stored in memory.

To summarise, inductive learning of word pronunciation is learning correspondences from known word-pronunciation pairs, and inducing from these

facts letter-phoneme correspondences for determining the pronunciations of new words, insofar as these new words are similar to known words. We have illustrated that the analogy principle can be combined with induction in inductive language learning.

The analogy principle as well as the use of induction in language are fundamental ideas that have been expressed in many different guises by influential linguists throughout the past century. We discuss landmarks in this tradition, as well as criticisms against it. On the basis of the linguistic debate, an updated, desired characterisation of inductive language learning is formulated.

Inductive language learning: landmarks

Until the 1960s, the most common opinion within linguistic research on language was that it was one of the once-mysterious human abilities that could be satisfactorily explained by generic learning methods. To provide a background for our problem statement, we highlight four linguists who, in different periods of the past century, have made explicit statements in favour of and against generic methods for language learning and processing: Ferdinand de Saussure, Leonard Bloomfield, Noam Chomsky, and Royal Skousen.

De Saussure: segmentation, classification, and analogy

In his book *Cours de linguistique générale*[2], the Swiss linguist Ferdinand de Saussure (1916) put forward the theory that linguistics (the study of language) naturally divides into a linguistics of *la langue*, viz. the language as a social medium, independent of the individual, and a linguistics of *la parole*, the individual's language. De Saussure (1916) argues that while *la parole* governs the way that individuals (learn to) generate and interpret speech, *la langue* represents the common, general knowledge of all individuals about the *signs* of the language, i.e., the common symbols of speech and writing used within a language community, and the relations existing between these signs.

Two relations exist between signs: (i) *syntagmatic* relations between signs at the same level; e.g., between letters in a word; (ii) *paradigmatic* (also referred to as *associative*) relations between signs at different levels, e.g., between letters and phonemes (De Saussure, 1916). Chomsky (Piatelli–Palmarini, 1980)

[2] The book was edited and published in 1916 after De Saussure's death in 1913 by his pupils C. Bally and A. Riedlinger, on the basis of (mostly handwritten) course notes by De Saussure (Harris, 1987).

points out that this dichotomy corresponds to two processes, viz. *segmentation* for determining the neighbourhood relations of (sequences of) linguistic symbols at the same level, and *classification* for determining the correspondences between (sequences of) linguistic symbols at different levels.

While syntagmatic and paradigmatic relations exist on the level of *la langue*, it is up to the individual's *parole* to learn both types of relations, e.g., to learn syntagmatic relations between letters, and to learn paradigmatic relations (correspondences) between words and pronunciations (De Saussure, 1916). For learning the relations, De Saussure (1916) argues, *analogy* is needed as the basic function. Our use of analogy in inductive language learning is in essence based on the Saussurian use of analogy.

Bloomfield: induction

While the role of analogy in language learning was expressed by De Saussure, the role of induction as the driving force behind discovering principles and regularities in language was advocated by the American linguist Leonard Bloomfield, who once stated "The only useful generalizations about language are inductive generalizations." (Bloomfield, 1933, p. 20). Bloomfield is renown foremost for his work on the development of standardisation methods for analysing (new) languages, and for being the founder of American structuralism (Robins, 1997). Bloomfield proclaimed radical ideas on the nature of language, which in his view was basically *behaviourist* (Gardner, 1987), i.e., emerging from learned responses to stimuli. Bloomfield argued that language is limited to what the speaker/hearer knows, through induction, about the relations between speech and writing symbols. Meaning, however strongly related to language, cannot be learned by straightforward induction, as it is much more complex than speech and writing; consequently, as Bloomfield (1933) argued, the study of meaning is not a part of linguistics.

Bloomfield's ideas, and those of behaviourist psychologists such as Skinner, dominated linguistics until the 1950s, when the American linguist Noam Chomsky launched a major attack against the Saussure–Bloomfield line of thinking about language.

Chomsky: abstraction levels, rules, principles and parameters

In Chomsky's view, it is a mistake to see language as a limited system of relations between relatively small sequences of linguistic symbols (words, pronunciations, letters, and phonemes, possibly some patterns of frequently

co-occurring words), of which the relations can be learned by induction (Chomsky, 1975). Language, Chomsky argued, not only incorporates speech, writing, and some superficial syntax, but incorporates meaning as well. He proposed a mathematical-logical formalisation of the relations between the *surface structure* of language (i.e., the way it appears in speech and writing), and the *logical form* of the meanings underlying utterances and sentences (Chomsky, 1957). Not being able in general to couple speech and writing to meaning directly, Chomsky originally introduced an intermediary level of *deep structure* between the levels of utterances and meaning. Utterances are converted into meaning and vice versa via deep structure and sets of generative and transformational grammar rules (Chomsky, 1957, 1965). Chomsky's recent work has reduced the concepts of deep structures and transformational and generative rules back to the framework of the *minimalist program* (Chomsky, 1995). The minimalist program aims to describe language by a system of principles and parameters, which is universal for all languages. The parameters, which are language specific, constitute the part of the language system that has to be learned. Once parameter values have been learned, the principle-part of the language system can couple articulatory language (speech) to concepts (meaning) via two *interface levels*: phonetic form and logical form (Chomsky, 1995). The mechanisms for deriving meaning from speech, and vice versa, via the interface levels, are kept as economical as possible.

Language learning is regarded by Chomsky as a specific, fast-paced discovery of parameters which determine the principles appropriate for the language being learned. Discovering the appropriate parameter settings cannot be driven by a generic learning process:

> "... I don't see any way of explaining the resulting final state [of language learning] in terms of any proposed general developmental mechanism that has been suggested by artificial intelligence, sensorimotor mechanisms, or anything else" (Chomsky, in Piatelli–Palmarini, 1980, p. 100).

Chomsky's ideas on generative and transformational grammars on the level of syntactic structures and logical form are not the focus of this thesis – we do not include aspects of meaning in our investigation of the word-pronunciation task, and we do not aim at falsifying Chomsky's claim expressed in the quote above. However, the ideas of Chomsky on generative and transformational grammars form a relevant background for our problem statement, since they have been transferred to linguistic phenomena at the

word level as well, in the fields of generative phonology and morphology, the linguistic domains in which word-level processes such as word pronunciation are investigated. The book *The Sound Pattern of English* (Chomsky and Halle, 1968), generally referred to as SPE, spawned a large amount of work on the discovery of generative and transformational models for isolated domains such as syllabic structure and stress patterns (Kenstowicz, 1993), and morphological structure (Sproat, 1992). The isolated investigation of the domains caused a change in how conversion processes such as word pronunciation were viewed, from a direct mapping from writing to pronunciation (as it was viewed upon by De Saussure, 1916), to a decomposed, step-wise conversion in which several *abstraction levels* are passed between the levels of writing and pronunciation.

Skousen: analogical modelling

A clear critique of the widely-accepted Chomskyan treatment of language and language learning in mainstream linguistics, is formulated by the American linguist Royal Skousen (1989). He argues that Chomsky is as much a structuralist as De Saussure and Bloomfield, and that Chomsky's critique was limited to the methodological assumptions put forward by the early structuralists (viz. discarding meaning, and adopting too blunt learning methods).

Controversial as it may be to classify Chomsky under the structuralists, the point that Skousen wants to make is that all mainstream-linguistic theories have assumed rules to be the only means to describe any aspect of language (Skousen, 1989). Indeed, De Saussure pointed out that analogy is employed to discover rules of associations and rules of syntagmatic relations, which exist in *langue* just as they exist in a domain with commonly-known rules such as chess (Harris, 1987).

Instead, Skousen argues for an inherently *analogical* approach to language and language learning. Explicitly deviating from De Saussure's vague notion of analogy, Skousen introduces a concrete definition of analogy that is not based on rules. This definition does not differentiate, as mainstream linguistic does, between regular language (i.e., language obeying rules), and irregular language (i.e., exceptions to the rules); rather, it treats language in an essentially unbiased way. Thus, to predict language behaviour, and to model language learning, all that is needed is a large data base of examples taken directly from language use, and a generically applicable method for analogical modelling which is inherently inductive, but avoids the induction of rules (Skousen, 1989; Derwing and Skousen, 1989).

Skousen advocates the analogical-modelling approach by highlighting the following advantages (Skousen, 1989):

(i) the analogical-modelling approach softens the undesirable all-or-none, static behaviour of rule-based approaches with respect to exceptions (which are only exceptions because, circularly, they do not fit the rules); in language, the differences between regular data and exceptions are usually graded, if they exist at all;

(ii) the approach is adaptive, rather than rigid, since it can adapt the learned model to new instances of the same task as they are presented to the analogical modelling algorithm;

(iii) it is much less complex than the discovery of rules, since it only involves access to a data base with a relatively straightforward learning principle;

(iv) analogical modelling induces models of which the behaviour often appears to be rule-governed on the outside: apparently, when rules can be employed to model the language task to be learned, they will be implicitly captured by analogical modelling.

Inductive language learning without explicit rules and abstraction levels

We agree with Skousen that inductive language learning should not be confined to the explicit induction of rules, but should be inherently analogical and unbiased with respect to language data. Our approach thus avoids the explicit induction of rules and, consequently, a distinction between regular and exceptional data. We investigate the possibility that the word-pronunciation task can successfully be learned by taking such an unbiased approach. With 'successful learning', we mean that the induced system will be able to pronounce new, unseen words with adequate accuracy. (The performance on new words is henceforth referred to as *generalisation accuracy*.) Moreover, we aim at demonstrating that such induction may be performed adequately without the explicit assumption of more than the levels of spelling and pronunciation.

A primary requirement for such an investigation is a criterion to describe adequacy of word-pronunciation generalisation accuracy. It would not suffice to apply a learning algorithm to the word-pronunciation task once and show that it can pronounce a good deal of new, unseen words correctly. More experiments are needed:

1. Systematic experiments are needed to incorporate linguistic-expert knowledge in the task definition, in order to measure the positive or negative effects of certain linguistically-motivated assumptions in the task definition. Can word pronunciation be learned without such assumptions in the task definition, or are certain assumptions (e.g., certain abstraction levels between the levels of writing and speech, cf. Section 1.1) essential for adequate accuracy?

2. We also need to compare the generalisation accuracy of the focus algorithm to that of other algorithms. An empirically important comparison would be between the focus algorithm and a well-chosen baseline accuracy.

Inductive language learning should therefore be performed in an *empirical* and *comparative* modus of research. Employing an empirical, comparative modus of research, it is possible to determine whether incorporating linguistic assumptions into the task definition makes the task easier or harder to learn. By varying the amount of linguistic assumptions in the task definition systematically, it is possible to chart in detail the influence of such assumptions on the overall learnability (generalisation accuracy) of the task being studied. Having compared the empirical results of the systematic experiments, one can truly test the claim of inductive language learning: that a specific language task (word pronunciation) can be learned with a generic inductive-learning method.

1.3 Machine learning: tools for inductive language learning

The method of our study of inductive language learning is to employ generic inductive-learning algorithms to language tasks. Such algorithms have been proposed and developed within the area of machine learning (ML). ML, a subfield of artificial intelligence (AI), investigates the computational realisation of learning. Our basic approach to ML is thus to see it as a resource of learning tools, among which the subset of tools desired for our study, i.e, the category of inductive-learning methods. This does not give right to the fact that ML is becoming a well-developed subfield of AI[3].

[3] Readers interested in the theoretical foundations of ML and important and current issues in ML are referred to Shavlik and Dietterich (1990), and Michalski, Carbonell, and Mitchell

Without going into detail about the state and developments of ML, it serves purpose to note that inductive learning is regarded as an important category of learning methods in *supervised* ML. Supervised ML concerns learning tasks in which the classes of the task to be learned are given, or in which the function to be approximated is given (Shavlik and Dietterich, 1990). In contrast, in *unsupervised* ML, the target classes or functions are not given and learning focuses on clustering or discovering classes or functions (Shavlik and Dietterich, 1990; Cheeseman *et al.*, 1988; Fisher, 1987). In this study we are concerned with supervised ML. Induction can be said to come naturally with generally-accepted definitions of supervised ML, which focus on the ability of learning systems to adapt continuously to new examples of the task to be learned (Simon, 1983; Michalski, 1993). As Simon (1983) states,

> *"Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time."* (Simon, 1983, p. 28, italics in original).

While other researchers have emphasised that learning should also include the ability to transfer the learned knowledge after being trained on a particular task to other related tasks (e.g., Carbonell, 1989), Simon's (1983) definition suffices for our purposes, since we investigate learning a single task at a time.

Learning by induction is the most obvious learning method to realise the desired adaptive behaviour expressed by Simon's definition, especially when the notion of "task" is taken to refer to real-world tasks. Real-world tasks are typified by exceptional, incomplete, or sometimes even inconsistent instances; the real-world task of word pronunciation, the object of our study, is no exception:

- **Exceptions** ('noise' would be the preferred term in ML) occur in word pronunciation with single words of which the pronunciation conflicts with a majority of similarly-spelled words. Examples in English are the words **have** and **having**, associated with the exceptional pronunciation /æv/ of the letters **av** pronounced differently in all other words with **av**. Other examples of (parts of) words of which the peculiar pronunciations

(1983), the consecutive volumes of that series (Michalski *et al.*, 1986; Kodratoff and Michalski, 1990; Michalski and Tecuci, 1994), and the recent text books by Langley (1996) and Mitchell (1997).

are better not used for determining the pronunciation of partly-similar words are the words **yacht**, **buoy**, **lieutenant**, and **quay**[4].

- **Incompleteness** can be found with homographs, i.e., pairs of similarly-spelled words of which the meaning determines the pronunciation. Incompleteness lies in the absence of indications of meaning in the spelling. Examples in English are **read**, **object**, and **suspect**.

- **Inconsistency** is pervasive in English, particularly on the level of pronouncing vowels: to quote the Dutch poet Charivarius (excerpted from *De Chaos*, Charivarius, 1922):

> Pronunciation —think of Psyche!—
> Is a paling, stout and spikey;
> Won't it make you lose your wits,
> Writing "groats" and saying groats?
> (...)
> Don't you think so, reader, rather,
> Saying lather, bather, father?
> Finally: which rhymes with enough —
> Though, through, plough, cough, hough, or tough?
> Hiccough has the sound of cup ...
> My advice is — give it up!

Inductive learning is inherently flexible in its best-guess approach to noisy, incomplete, and inconsistent examples typical of real-world tasks, although the inherent flexibility does not guarantee that any task can be learned (Breiman, Friedman, Ohlsen, and Stone, 1984; Quinlan, 1986; Kibler and Aha, 1987; Aha, Kibler, and Albert, 1991; Schaffer, 1993, 1994.). Thus, our aim to employ inductive learning of word pronunciation, a real-world task, is in accordance with the general opinion in ML identifying inductive learning methods to be well-suited in principle. An inventory of available inductive-learning algorithms which can learn real-world tasks will produce a selection of specific algorithms with which we will conduct our study.

[4] The four examples are loan words: **yacht** and **buoy** stem from Dutch, and **lieutenant** and **quay** from French; loaning words from other languages plays a significant role in introducing peculiar pronunciations of (parts of) words.

1.4 Problem statement

We have established the following:

- There is an apparent contrast between the view of traditional structuralist and Chomskyan linguistics on the possibility of inductive-learning methods being able to learn language tasks such as word pronunciation. Traditional structuralists claim that it is possible or even essential; Chomskyan linguists claim that it is impossible, since the abstraction levels and rules assumed necessary cannot be learned autonomously by generic learning methods such as induction.

- It has been proposed (Skousen, 1989) that both the traditional structuralists and mainstream Chomskyan linguists are incorrect in assuming language to be governed by rules, and language learning to consist of the discovery of rules. Language learning may well be realised by assuming purely data-based analogical modelling as the learning method, which is equivalent to inductive learning methods that do not explicitly induce rules (cf. Daelemans, 1995; Daelemans, 1996b for similar arguments). We adopt this method of inductive language learning, and extend it to a comparative testbed method of learning, in which the explicitness of linguistic assumptions in the task definition can be systematically varied.

- Inductive-learning algorithms have been developed within ML; moreover, theories of ML point to inductive learning constituting a set of well-suited methods for learning to perform real-world tasks, since inductive-learning methods are inherently flexible towards exceptions, incompleteness, and inconsistencies which typically occur in real-world tasks.

Furthermore, we have been collecting, since 1991, pieces of evidence that inductive-learning algorithms can learn specific language tasks (Daelemans, 1995), among which subtasks of word pronunciation, such as hyphenation (Daelemans and Van den Bosch, 1992a), syllabification (Daelemans and Van den Bosch, 1992b), grapheme-phoneme conversion (Weijters, 1991; Van den Bosch and Daelemans, 1993; Van den Bosch, Content, Daelemans, and De Gelder 1995), and stress assignment (Gillis, Durieux, and Daelemans, 1993; Daelemans, Gillis, and Durieux, 1994a). These studies involved different algorithms, and differently-sized data sets of different languages. Although

statements can only be made properly for each study separately, the studies consistently demonstrate that inductive-learning algorithms that do not explicitly learn rules or abstraction levels can learn the language tasks with an accuracy ranging form adequate to surprisingly high (cf. Daelemans, 1995).

With this background, we formulate our problem statement:

> *Can generic inductive-learning algorithms learn to pronounce written words with adequate generalisation accuracy, even when the task definition assumes none of the abstraction levels assumed necessary by linguistic experts?*

To answer this question the thesis is organised stepwise; each chapter describes one step. The next section describes the sequence of steps taken.

1.5 Thesis outline

The thesis is organised as follows. Chapter 2 introduces (i) the selected inductive-learning algorithms, (ii) linguistic theories and assumptions underlying word pronunciation, and (iii) the methodology adopted for the study. The introductory nature allows for readers familiar either with ML and inductive-learning algorithms, or with (computational) morphology and phonology, to skip the first and second section, respectively. The third section on methodology describes in detail how the empirical part of the study, described in Chapters 3 to 6, is performed systematically with fixed methods.

The "body" of the thesis is formed by the chapters describing the empirical results of our experiments, viz. Chapters 3 to 6, organised according to the systematic variations applied to the definition of the word-pronunciation task. Over the chapters, the amount of linguistic assumptions incorporated in the task definition is gradually decreased from *considerable* to *none*.

In Chapter 3, the linguistic assumptions concerning five different abstraction levels discerned within word pronunciation are translated into five separate subtask definitions: the word pronunciation task is dismantled into (i) morphological segmentation, (ii) grapheme-phoneme alignment, (iii) grapheme-phoneme conversion, (iv) syllabification, and (v) stress assignment.

In Chapter 4, these five subtasks are sequentially coupled as modules in modular systems. The sequential order is based on the sequential order of subtasks in two existing systems for text-to-speech conversion. In the same chapter, the linguistic assumptions on the modularisation of the task

are weakened by testing equivalent modular systems with less modules, i.e., by integrating pairs of subtasks in single subtasks.

In Chapter 5, the idea of sequential modularisation is abandoned, and instead different parallel modularisations are tested. In these systems, linguistically-motivated subtasks are performed in parallel, under the assumption that they can be performed independently. The number of parallel modules is varied, and includes the bottom-line case of a single module, which performs word-pronunciation in one single pass (thus, without the assumption of any abstraction level).

In Chapter 6, the idea is elaborated that the word-pronunciation task can also be decomposed by purely data-driven criteria, rather than by linguistic criteria as in the parallel modular systems described in Chapter 5. It explores three *gating* systems, in which the word-pronunciation examples are automatically divided into two non-overlapping subsets. The underlying idea is that a successful gating method may lead to the decomposition of the word-pronunciation task into partial word-pronunciation tasks with essentially different characteristics, by which each partial task can be learned better than the word-pronunciation task as a whole.

A summary and discussion of the empirical results is given in Chapter 7. The chapter provides analyses of characteristics of the word-pronunciation task enabling inductive learning to learn it, presents an overview of related research, and identifies the novel contributions of our study, and provides a list of indications for future research. Chapter 8 summarises the conclusions drawn from the study.

Notes on typesetting

The text in the thesis is typeset in the Palatino font family. Words, parts of words, or letters that serve explicitly as examples are typeset in the bold Helvetica font, e.g., **example**, **ple**, and **p**. Exemplified pronunciations follow the guidelines of the International Phonetic Alphabet, as proposed by the International Phonetic Association (1993), and are always preceded and followed by the '/' mark, e.g., /prənʌnsɪeɪʃən/. A list of all letters and phonemes used is given in Appendix A.

Chapter 2

Inductive learning of word pronunciation

In this thesis we investigate the ability of generic inductive learning of word pronunciation to attain adequate generalisation accuracy. For this investigation four requirements are assumed, viz. (i) the availability of implemented inductive-learning algorithms, (ii) a definition of the word-pronunciation task, (iii) the availability of instances of the task, and (iv) a well-defined experimental methodology for applying the learning algorithms to the task.

The four requirements are treated in the Sections 2.1, 2.2, 2.3, and 2.4 of this chapter. In Section 2.1, three groups of generic inductive-learning algorithms are identified; they are in principle suited for learning complex tasks. For each group representative algorithms are selected and described. Throughout the section, a small example word-pronunciation task (viz. the pronunciation of the **k** in English words containing the two-letter string **kn**) is used to demonstrate the functioning of the algorithms.

In Section 2.2, a description is given of the abstraction levels assumed necessary by developers of mainstream word-pronunciation systems to perform the task of word pronunciation, and definitions are formulated of the word-pronunciation task and subtasks assumed by word-pronunciation system developers, inspired by linguistic theory. In Section 2.3, details are provided on the data base from which we acquire the word-pronunciation material used throughout the thesis.

Section 2.4 describes the experimental methodology. It provides a description of the regimen under which all experiments are performed.

2.1 Inductive-learning algorithms

Selecting a set of inductive-learning algorithms to be used in our experiments, we define three conditions to which candidate inductive-learning algorithms must adhere:

1. The inductive-learning algorithms must not distinguish explicitly between regular data and exceptions and use this distinction to ignore certain instances on the basis of their exceptionality. With this condition we bring about the argument adopted from Skousen (1989), that inductive language learning should treat all language data in an unbiased analogical manner, and avoid any distinction between (rule-governed) regular data and exceptions that do not fit the rules (cf. Section 1.2).

2. The selected inductive-learning algorithms must be able to learn complex tasks, that contain exceptions, incompleteness, and inconsistencies, and for which large amounts of different instances are available, such as the language task investigated here (cf. Section 1.3).

3. The algorithms must offer maximally different approaches to inductive learning and to classification. This is to ensure that balanced statements can be made as regards the advantages and disadvantages of the tested algorithms (i.e., that statements can be made on the relations between performance differences of algorithms and differences in their learning approaches).

The first condition excludes two groups of inductive-learning algorithms. First, it excludes algorithms in which learning involves removing exceptions from the data until it represents only regular (rule-governed) or prototypical data. Such algorithms generally employ a computational approximation of the *minimal description length principle* (Rissanen, 1983). By employing utility thresholds, they attempt to minimise the number of rules needed to cover the data. Exceptions that do not fit the rules and are infrequent (e.g., below a certain threshold) are ignored (i.e., no rules are generated for too small amounts of data). Examples of such algorithms that ignore exceptions by *pruning* them, or by *editing* the data, are the rule-induction algorithms CN2 (Clark and Niblett, 1989), FOIL (Quinlan, 1993), C4.5 with pruning (Quinlan, 1993), C4.5-RULES (Quinlan, 1993), and RIPPER (Cohen, 1995). An example algorithm in which the data is reduced by explicitly removing exceptions (noisy instances) but in which no rule induction is performed, is the instance-based

algorithm IB3 (Aha, Kibler, and Albert, 1991); *'edited k*-nearest neighbour classifiers' is a common term denoting instance-based learning algorithms that remove noisy instances from memory during learning (Wilson, 1972; Voisin and Devijver, 1987).

Second, the first condition excludes algorithms which assume expert knowledge to be employed for representing the data. This can be said to concern inductive logic programming (ILP) algorithms (Lavrac and Džeroski, 1994; Lavrac *et al.*, 1996) and typical case-based reasoning (CBR) approaches (Kolodner, 1993). Both approaches are capable of handling many types of inputs (e.g., variable-length feature-value vectors, horn clauses, or acyclic graphs), while also needing generally the incorporation of expert (domain) knowledge to restrict the functioning of ILP and CBR algorithms within the boundaries of computational feasibility (Lavrac and Džeroski, 1994; Kolodner, 1993). The inclusion of expert knowledge is what our approach aims to avoid, as well as explicitly inducing rules (cf. Section 1.2).

We do not claim that the algorithms excluded by the first condition are unable to learn language tasks. It would constitute a separate line of research to investigate their learning abilities and computational feasibilities; we are not concerned with these issues here. Our approach attempts to take an inherently unbiased approach towards language data, placing it opposite to the mainstream linguistic claim that rules are necessary (cf. Section 1.2).

We identify three groups of algorithms not excluded by the first condition. As will be demonstrated in this section, the algorithms in these groups do not differentiate between regular and exceptional data in order to ignore the exceptions:

1. connectionist learning;
2. non-edited instance-based learning; and
3. non-pruning decision-tree learning.

The learning abilities of these three groups of algorithms are relevant current issues in machine-learning research. For all three groups it has been argued theoretically that they can learn complex tasks, in principle, under certain assumptions: Rosenblatt (1958), Hornik, Stinchcombe, and White (1989), White (1990), and Faragó and Lugosi (1993) for connectionist learning; Cover and Hart (1967) and Aha *et al.* (1991) for instance-based learning, and Hunt (1962), Hunt, Marin, and Stone (1966), and Quinlan (1986) for decision-tree learning. The three groups of algorithms thus adhere to the second condition.

The performance, mostly in terms of generalisation performance, of inductive-learning algorithms from all three groups has been investigated on complex language tasks for which large amounts of instances were available:

1. **Connectionist-learning** algorithms have been applied successfully to English grapheme-phoneme conversion trained on the 20,000-word NETTALK corpus (Sejnowski and Rosenberg, 1987); to English word hyphenation using a 89,000-word corpus (Van den Bosch *et al.*, 1995); and to Dutch hyphenation and syllabification using a 70,000-word corpus (Van den Bosch and Daelemans, 1992). For all tasks, the connectionist-learning algorithm involved, viz. back-propagation (Rumelhart, Hinton, and Williams, 1986), was claimed to attain high accuracy in terms of generalisation accuracy at least comparable to standard linguistic rule-based approaches.

2. **Instance-based learning** algorithms were also applied to a wide range of language tasks, e.g., the NETTALK task (Weijters, 1991; Van den Bosch and Daelemans, 1993); grapheme-phoneme conversion in French and Dutch trained on 20,000-word corpora (Van den Bosch and Daelemans, 1993); and morphological segmentation trained on a 75,000-word corpus (Van den Bosch, Daelemans, and Weijters, 1996).

3. Applications of **decision-tree learning** algorithms to the NETTALK task were demonstrated to be successful in terms of generalisation accuracy, and to be equally or more accurate than back-propagation (Dietterich, Hild, and Bakiri, 1995; Dietterich and Bakiri, 1995; Van den Bosch *et al.*, 1995). The same was demonstrated for Dutch grapheme-phoneme conversion, trained on a 70,000-word corpus; this decision-tree learning algorithm was demonstrated to outperform a state-of-the-art rule-based grapheme-phoneme-conversion model (Van den Bosch and Daelemans, 1993).

The third condition is fulfilled by the three groups of algorithms differing maximally in two dimensions of inductive learning, viz. (i) *learning effort* and (ii) *classification effort*:

- The **learning effort** dimension ranges from a minimal to a maximal computational effort put in inducing information from task instances

presented during learning. A minimal learning effort occurs in algorithms which simply store all training material in memory, without further processing of this material. Maximal learning effort occurs in algorithms in which learning takes a potentially infinite time, and in which complex (e.g., recursive) indexing, abstraction, or compression processes are applied to the training material.

- The **classification effort** dimension ranges from minimal to maximal computational effort invested in the retrieval of stored information to classify instances of the trained task. Minimal classification effort occurs when classification of new instances is performed in a deterministic, non-backtracking, single-pass process. Maximal classification effort occurs when classification of new instances involves a brute-force search through all information stored in memory.

Figure 2.1 visualises the two dimensions as x-axis and y-axis, respectively, spanning up a two-dimensional space in which the three groups of algorithms can be found. It illustrates that each of the three groups of algorithms is located near the end of at least one of the two dimensions.This visualisation is schematic and should be interpreted as rough estimates of practical computational effort, such as processing time (e.g., in cpu cycles) or memory (e.g., in bytes). In addition to these rough estimates, we provide asymptotic complexity analyses for the three groups of algorithms and report on memory usage and processing time in detail in Chapter 7. Estimating the coordinates of algorithm groups in Figure 2.1 in practical terms is oblivious of the fact that algorithms from one group may be implemented as algorithms from another group; e.g., instance-based learning and decision-tree learning may be implemented in connectionist networks (e.g., Ivanova and Kubat, 1995). Moreover, the two-dimensional figure ignores other dimensions relating to theoretical (e.g., worst-case, lower-bound, upper-bound) complexities, limitations, and abilities.

The learning effort invested by connectionist learning is potentially maximal. The connectionist-learning process is a set of repetitive, complex matrix operations, which generally need to be constrained by stopping criteria. Moreover, connectionist learning constitutes a complex encoding and decoding of symbolic information through non-symbolic intermediary representations. The classification effort in connectionist-learning algorithms is moderate (in terms of processing); it involves the same encoding and decoding process as the learning component, but it lacks the computational burden of the costly repetitive matrix operations employed during learning.

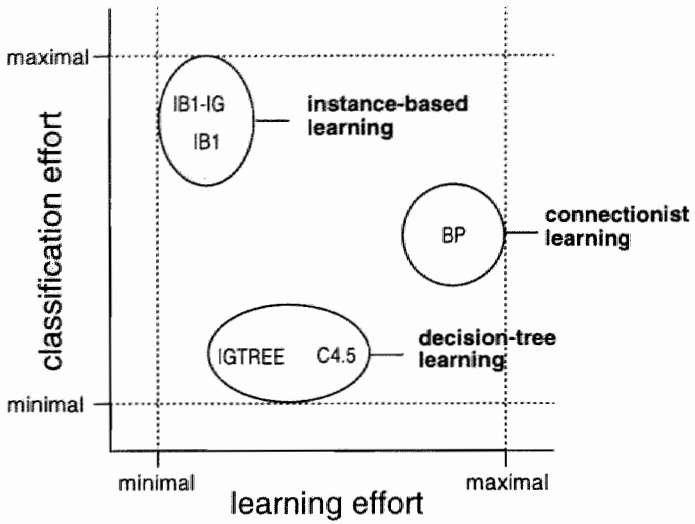


Figure 2.1: Schematic visualisation of the position of the three inductive-learning-algorithm groups within the two-dimensional space bounded by the learning-effort dimension and the generalisation-effort dimension.

Instance-based learning is characterised by a near-minimal learning effort. Learning involves storing instances straightforwardly in memory in a data base of instances. Classification effort, in contrast, is maximal, involving a brute-force search through the constructed instance base, investigating all information present in the data base. In Section 1.2, we characterised this type of inductive learning as 'on demand' (p. 6): classification of new instances is delayed until these instances are actually presented.

Decision-tree learning is characterised by a moderate learning effort. Learning is relatively fast and deterministic, but involves a recursive process of indexing and compressing an instance base into a decision tree. Classification effort in decision trees is near minimal; it involves a simple, deterministic, non-backtracking pass through the decision tree.

Altogether, Figure 2.1 displays a good dispersion of the three algorithm groups within the two-dimensional space bounded by the two dimensions. Thus, the third condition is met.

The unoccupied areas in Figure 2.1, viz. the upper right corner and the lower left corner, indicate the position of groups of inductive-learning algorithms not meeting the second condition for candidate learning algo-

rithms. Algorithms displaying minimal learning effort as well as minimal classification effort (i.e., in the lower left corner) can be expected to display low generalisation accuracy when trained on complex tasks. Alternatively, algorithms displaying high learning effort as well as moderate to high classification effort (i.e., in the upper right corner) run the risk of being infeasible in terms of processing time, memory requirements, or both, when applied to complex tasks with large amounts of training material. We have reason to assume that Skousen's (1989) analogical-modelling algorithm is an example of the latter type of algorithm. The examples and analyses of the functioning of the algorithm presented by Skousen (1989) show that the algorithm behaves computationally rather costly with respect to memory and learning (Skousen, 1989, p. 51). Daelemans, Gillis, and Durieux (1994b) show that the analogical-modelling algorithm is exponential in the number of features. The limitations of the algorithm forced Skousen in a number of examples to introduce linguistic expert knowledge (albeit somewhat low-level) to reduce the number of input features (Skousen, 1989), which is what we aim to avoid in our study. Skousen claims that in the absence of readily-available massive parallel processing machines, his algorithm will suffer from these limitations (Skousen, 1989, p. 52).

Having selected the groups of inductive-learning algorithms, we need instantiations of learning algorithms within these groups to be employed in our experiments. For each of the three groups we select one representative, well-known algorithm, viz. back-propagation (BP) (Rumelhart *et al.*, 1986) for connectionist learning, IB1 (Aha *et al.*, 1991) for instance-based learning, and C4.5 (Quinlan, 1993) for decision-tree learning. Subsequently, we add to each of the latter two algorithms one algorithmic variation which was demonstrated earlier in empirical research to perform well on complex language tasks. First, IB1-IG (Daelemans and Van den Bosch, 1992a; Daelemans, Van den Bosch, and Weijters, 1997a) is selected as being a variation of instance-based learning in IB1. IGTREE (Daelemans *et al.*, 1997a) is selected as a variation of decision-tree learning in C4.5 (although it can be seen as an optimisation of IB1-IG as well, cf. Daelemans *et al.*, 1997a).

In Subsection 2.1.1, we introduce the task of pronouncing **kn** in English words, a subtask of word pronunciation with which the functioning of the algorithms is illustrated. BP is described in Subsection 2.1.2; IB1 and IB1-IG are described in Subsection 2.1.3, and Subsection 2.1.4 describes C4.5 and IGTREE.

2.1.1 A sample task: Pronouncing kn

As an example of a small subtask of word pronunciation we introduce the **kn** task, with which we will demonstrate the functioning of the inductive-learning algorithms in Subsections 2.1.2 to 2.1.4. The **kn** task is defined as *determine the pronunciation of the* **k** *in* **kn** *in a word containing this two-letter string*. The pronunciation of the letter combination **kn** in English words is irregular. When collecting examples of English **kn** words, one finds that the **k** in **kn** is generally not pronounced: e.g., as in **knock**, **knife**, **knight**, **knot**, and **unknown**. Some examples, however, show that when the **k** and the **n** belong to different parts (*morphemes*) of the word, the primary rule is overruled and the **k** is pronounced: e.g., as in **banknote**, **meekness**, and **weeknights**. In other examples, the **k** is pronounced when it is preceded by a **c**, as in **picknick** or **cockney**[1]; a morpheme boundary does not always occur in these cases.

CELEX (Van der Wouden, 1990; Burnage, 1990) is a collection of lexical databases of English, Dutch, and German words. It is used throughout this thesis as the source for word-pronunciation data. In CELEX, we found 270 English words containing the two-letter string **kn**[2]. The **k** is not pronounced in 223 of these words (83%).

For experimenting with the five learning algorithms learning the **kn** task, the 270 words with their specific pronunciations of the **k** in **kn** need to be formatted as *instances* of the **kn** task. An instance is a combination of a representation of the word and its associated classification, e.g., a representation of the word **weeknight** and its associated classification 'the **k** is pronounced'. Once a set of instances is available, the algorithms can perform their specific type of inductive learning. Instances on which inductive learning is based are referred to as *training instances*. After inductive learning has been completed, the algorithm can classify instances for which the associated classification is *unknown*. Instances with associated classifications unknown to the learning algorithm are referred to as *test instances*.

The five learning algorithms all require that the instance representations of the words be represented by vectors of *feature-value* pairs. The features are the word positions, and the values are the letters. Moreover, the length of these vectors needs to be fixed. This implies that the words themselves cannot be used as instances, since they have different numbers of letters, and that a sizing method is needed. For the case of the **kn** task, we place a virtual

[1] In these cases in which **k** is preceded by a **c**, in which we consider the **k** to be pronounced, it is actually the letter group (or *grapheme*) **ck** that is pronounced as /k/.

[2] Upper-case **K** and **N** are counted as, and converted to, lower-case **k** and **n**.

#	left			k	n	right			classification
1	o	r	e	k	n	o	w	n	/-/
2	_	_	_	k	n	e	e	c	/-/
3	_	_	_	k	n	i	t	t	/-/
4	s	i	c	k	n	e	s	s	/k/
5	o	r	e	k	n	e	w	_	?

Table 2.1: Five examples of instances of the **kn**-task. The instances are generated by windowing of the words **foreknown**, **kneecaps**, **knitting**, **sickness**, and **foreknew**, respectively. The final word is the test word.

window on the word, positioning **kn** in the centre, and including three letters to the left of **kn**, and three letters to the right of **kn**. Each word is thus converted into a vector of eight feature-value pairs. The first three feature-value pairs represent the three left-context letters to **kn**, the fourth and fifth feature-value pairs represent **k** and **n**, respectively, and the final three feature-value pairs represent the three right-context neighbour letters. Since the window of the windowing method may extend beyond word boundaries, e.g., when **kn** is at the beginning of a word (cf. instances 2 and 3 in Table 2.1), an extra feature value is introduced, viz. '_'. This symbol acts as a feature value as any other letter.

The instances are divided over a training set and a test set. We create a simple test set containing only one instance, **oreknew**_, derived from the word **foreknew**; the training set contains all other 269 instances. Table 2.1 displays the test instance **oreknew**_ and four example training instances. The classification of instances of the **kn** task is encoded by two labels, viz. /-/, denoting that the **k** is not pronounced, and /k/, denoting that the **k** is pronounced. The question mark in the classification field of the test instance **oreknew**_ indicates that the classification of this instance is not known to the learning algorithms. Of course, the correct classification of this instance is /-/; this knowledge will be compared to the actual classification output of the algorithms in Subsections 2.1.2 to 2.1.4.

The locality assumption

Our method for constructing fixed-size feature-value representations of words ignores the fact that **kn** can be preceded (as in **sleeping-sickness**) and followed

(as in **knitting-machines**) by more than three letters: we deliberately constrain the context included in each instance. We do this because we assume that the eight letters represented in the instances contain *sufficient* information for the algorithms to learn the classifications of all training instances successfully as well as to classify test instances optimally. We assume that the **kn** task can be learned using only *local* information immediately surrounding **kn**. This is a strong assumption; we will henceforth refer to it as the *locality assumption*. It will be maintained throughout this thesis, for different types of word-pronunciation tasks and subtasks, and will be critically discussed in Chapter 7.

2.1.2 A connectionist-learning algorithm

The study of connectionist systems, or artificial neural networks (ANNs) (Rumelhart and McClelland, 1986; Anderson and Rosenfeld, 1988), is at least as multi-disciplinary as the field of machine learning, with which it partly coincides. ANNs are systems composed of highly interconnected networks of nonlinear computing elements, whose structure is inspired by early knowledge of biological neural systems. In these systems, learning rules govern the process of changes in the connections between the computing elements in the network (analogous to the synaptic connections between brain cells).

Back-propagation

For our experiments, we employed the well-known BP learning rule (Rumelhart *et al.*, 1986)[3]. We provide a description of BP and illustrate its functioning by applying it to the **kn** task. BP is a learning algorithm designed for a specific type of ANNs, viz. *multilayer feed-forward networks* (MFNs). MFNs are composed of interconnected computing elements (henceforth referred to as *units*) organised in three or more layers: an input layer, an output layer, and one or more hidden layers. In an MFN successfully trained with BP on a classification task, the input layer represents the feature values of instances, and the output layer represents the class to which the instance belongs, i.e., its associated classification. All units of a layer are connected to all units of the adjacent layer(s), i.e., each adjacent pair of layers is fully connected.

[3] Although 'BP' was coined by Rumelhart *et al.* (1986) and although the BP learning algorithm was popularised due to the wide acceptance of their description, it was proposed earlier by Werbos (1974) and reinvented by Parker (1985).

Each unit has an *activation*, i.e., a real value between 0 and 1. Activations of input units represent feature values of instances; activations of hidden units and output units are determined via *activation propagation*. During this process, each input unit's activation is propagated over all connections from that input unit to all hidden units. Propagated activations are multiplied by the *weight* of the connection, which is a real-valued number between $-\infty$ and ∞[4]. The net input of a hidden unit is the sum of all weighted activations it receives over its connections with all input units. The activation of the hidden unit is then computed by converting the net input to a number between 0 and 1. We use a sigmoidal function to perform this conversion (cf. Rumelhart *et al.*, 1986). The equation for this function is given in Appendix B as Eq. B.2. Activations of output units are computed analogously (output units receive their net input from the hidden units).

We now turn to the role of the back-propagation learning algorithm (BP). Ideally, the activation of input units representing the feature values of an instance leads via activation propagation to an activation of output units representing the associated classification of the instance. The accuracy of this classification is crucially dependent on the values of the connection weights. The goal of BP is to find values of the connection weights enabling such mappings.

BP is invoked by entering a *training phase* of repeated *cycles*. In one cycle, all training instances are presented to the MFN. For each instance, the input unit activations are set to encode the instances' feature values, and activation propagation determines the activation of the output units. When the difference between an output unit's desired and actual activation is larger than a threshold value t, BP changes the weights of the connections from all hidden units to this output unit according to the *generalised delta rule* (Rumelhart *et al.*, 1986): for each connection to the output unit, a *weight change* is computed which is a multiplication of (i) the *error* of the output unit, (ii) the activation of the hidden unit at the other end of the connection, and (iii) a value representing the *learning rate* parameter, set to 0.1 in our experiments; to the resulting value, a portion of the weight change at the previous instance presentation is added to the current weight change. The *momentum* parameter (Rumelhart *et al.*, 1986) determines how large this portion is (it is set to 0.4 in our experiments). The threshold value t mentioned above is an addition to the standard definition of BP; it results in an automatic reduction of the number of weight changes as compared to standard BP, and results in reductions in learning time, with our

[4] In our experiments, the absolute values of the weights cannot exceed 15.0.

typical data, by about a factor 2 to 3. In Appendix B, a detailed description is given of the generalised delta rule for connection-weight changing between all layers in the MFN.

After each cycle, a *network error* is computed, which is a cumulative error over all instances. This error can be used to determine whether training can be stopped. We define the network error as the percentage of training instances classified incorrectly during one cycle. The training phase is halted (i) when the network error does not differ from the network errors measured during the previous four cycles (the *halting patience* parameter) within a margin of 0.25% (the *patience threshold* parameter), or (ii) when the error reaches 0.0%. If neither of these two conditions is met, another cycle is entered.

Upon completion of the training phase, the MFN can be used to classify test instances in a *test phase*. This is done by performing activation propagation after the input units are assigned activation values representing the feature values of the test instance. All output units receive a certain activation; the output unit with the highest activation is taken as the classification of the instance.

BP can learn complex tasks when at least two conditions are met: (i) an adequate network architecture must be chosen, specifying the appropriate number of hidden layers and number of units per layer, and (ii) the learning rate and momentum parameters must be set to appropriate values. No sound rules are available for determining the optimal parameter values. Only heuristics and rules-of-thumb are available (Moody, 1992; Weigend and Rumelhart, 1994; Lawrence, Giles, and Tsoi, 1996); moreover, only limited freedom is allowed to search for appropriate numbers and values by performing experiments with BP on MFNs, when the results from these experiments should be statistically sound (cf. Subsection 2.4.2).

When applying BP to the **kn** task, one must first establish a coding scheme for translating feature values of instances into input unit activations, and for translating instance classifications into output unit activations. We choose to assign each individual feature value one unit in the input layer, and to assign each individual classification one unit in the output layer: absence and presence of input feature values and output classifications is represented by activation values of 0 and 1 for input units, and 0.1 and 0.9 for output units[5], respectively. The instances of the **kn** task are characterised by eight features $f_1 \ldots f_8$. In the 270 words containing **kn**, different letters occur at

[5] Output units are trained to have activation values 0.1 and 0.9, rather than 0 and 1, since the slight offset of 0.1 speeds up learning by BP (Sarle, 1997).

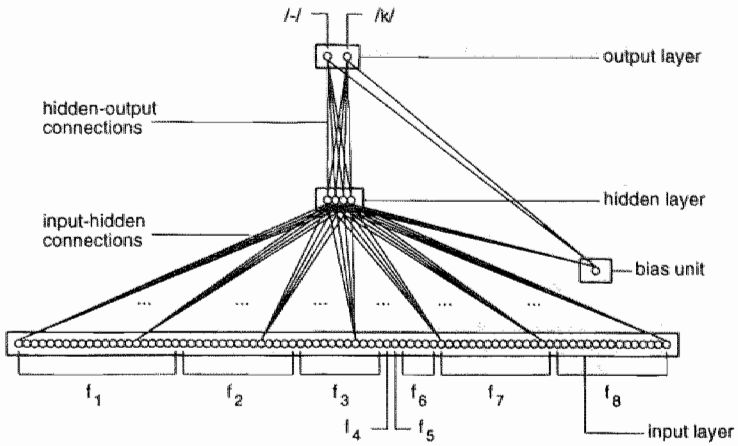


Figure 2.2: Visualisation of the MFN used in the application of BP on the **kn** task. The groups of input units representing the feature values of each of the features are indicated by $f_1 \ldots f_8$. Not all connections between input units and hidden units are shown.

each of these features, albeit not all letters of the alphabet: for example, at f_6, i.e., the feature representing the letter position immediately to the right of **kn**, only the letter values **a**, **e**, **i**, **o**, and **u** occur. All occurrences of feature values in the eight features add up to an input layer of 84 units. The output layer contains two units, one for representing /-/, and one for representing /k/. We choose an MFN architecture with one hidden layer containing four units. This choice is arbitrary. The settings lead to an MFN containing a total of $84 + 4 + 2 + 1 = 91$ units (the last unit in the addition is the bias unit, cf. Appendix B), and $(84 \times 4) + (4 \times 2) + 4 + 2 = 350$ connections. This MFN is displayed schematically in Figure 2.2 (not all connections are shown for the purpose of graphical clarity).

Applied to the **kn** task, the training phase takes 17 cycles, after which all training instances are classified correctly and the stopping criterion is met. Figure 2.3 displays the network error curve, i.e., the percentage classification error on training instances during the training phase. As can be seen in Figure 2.3, the amount of error decreases fairly smoothly until the ninth cycle, after which the error does not decrease for three cycles. During this period, the network already classifies 266 out of the 269 instances correctly; only the

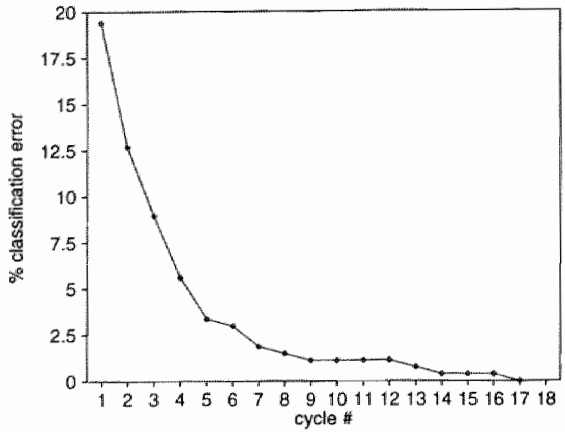


Figure 2.3: Network error curve of BP-training of the **kn** task. The network is trained for 17 cycles, after which the error on the training set reaches 0.0%.

instances **topknots**, **lipknots** (from **slipknots**), and **__kness** (from **Knesset**[6]) are classified incorrectly. After the fourteenth cycle, **topknots** and **lipknots** are classified correctly, and during the final three cycles the correct classification of **__kness** (/k/) is learned.

When the test instance **oreknew**_ is presented to the network, activation propagation leads to the following activation values of the two output units: 0.57 for the unit representing the /-/-class, and 0.43 for the unit representing the /k/-class. We take the classification to be the label of the output unit with the highest activation, viz. /-/. Thus, BP can reproduce the classifications of the training instances of the **kn** task successfully after training, and classifies the test instance **oreknew**_ correctly as /-/.

2.1.3 Two non-edited instance-based-learning algorithms

Instance-based learning is partly based on the hypothesis that performance in classification tasks may be successfully based on analogical reasoning, i.e., on computing the analogy (i.e., similarity, cf. Section 1.2) between new instances and stored representations of instances encountered earlier, rather

[6] **Knesset**, a proper name from Hebrew, is the only word in English beginning with **kn** in which the **k** is pronounced. Upper-case characters are converted to lower-case (Subsection 2.1.1.)

than on the application of rules abstracted from earlier experiences (Aha *et al.*, 1991; Aha and Goldstone, 1992; Daelemans, 1995). Instance-based learning is sometimes referred to as *lazy learning*, due to the minimal effort put in the learning process. We characterised this type of inductive learning earlier as 'on demand' (p. 6). Synonymous terms found in the literature are exemplar-based, case-based, and memory-based learning or reasoning (Stanfill and Waltz, 1986; Kolodner, 1993).

We describe two instance-based learning algorithms that do not employ editing of the data to remove exceptions, viz. IB1 and IB1-IG, descendants of the k-nearest neighbour algorithm (Cover and Hart, 1967; Devijver and Kittler, 1982; Aha *et al.*, 1991).

IB1

IB1 (Aha *et al.*, 1991) constructs a data base of instances (the *instance base*) during learning. An instance consists of a fixed-length vector of n feature-value pairs, an information field containing the classification(s) of that particular feature-value vector, and an information field containing the occurrences of the classification(s). The two information fields thus keep track of the occurrence of the feature-value vector in the training set, and the number of associated classifications of the vector occurrences. After the instance base is built, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance* between the new instance X and the memory instance Y, $\Delta(X,Y)$, using the function given in Eq. 2.1:

$$\Delta(X,Y) = \sum_{i=1}^{n} W(f_i)\delta(X_i,Y_i), \tag{2.1}$$

where $W(fi)$ is the weight of the ith feature (in IB1, this weight is equal by default for all features), and $\delta(x_i,y_i)$ is the distance between the values of the ith feature in the instances X and Y. When the values of the instance features are symbolic, as with our tasks, a simple distance function is used (Eq. 2.2):

$$\delta(X_i,Y_i) = 0 \; if \; X_i = Y_i \; else \; 1. \tag{2.2}$$

The classification of the memory instance Y with the smallest $\Delta(X,Y)$ is then taken as the classification of X. This procedure is also known as 1-NN, i.e., a search for the single nearest neighbour, the simplest variant of k-NN (Devijver and Kittler, 1982).

We have made three additions to the original IB1 algorithm (Aha *et al.*, 1991) in our version of IB1 (Daelemans and Van den Bosch, 1992a; Daelemans *et al.*, 1997a). First, when a single best-matching instance is found associated to more than one classification, our version of IB1 selects the classification with a highest occurrence in the instance's class distribution. Second, when a single best-matching instance is found associated with more than one classifications with equal occurrences, the classification is selected with a highest overall occurrence in the training set. Third, in case of more than one best-matching memory instance, the classification is selected with a highest occurrence as summed over the classification distributions of all best-matching instances. On finding more than one best-matching instances, this function merges the classification distributions of all best-matching instances and selects the classification with the highest overall occurrence in this merged distribution (Daelemans *et al.*, 1997a).

In our implementation of IB1, we have optimised classification by dividing it into two phases. First, IB1 searches for duplicate instances (i.e., instances in the instance base with totally identical feature values as the test instance) in the instance base using an alphabetised index. Since duplicate instances are necessarily best-matching, the search can be halted when duplicates are found (which, for our data, occurs relatively frequently; cf. Chapter 3 and onwards); their classification determines the classification of the test instance. If no duplicates are found, IB1 enters the second phase in which simply all instances in the instance base are matched against the test instance.

When IB1 is applied to the **kn** task, it builds an instance base containing 134 instances, i.e., all uniquely sorted instances accompanied by their class labels and class occurrence counts. Figure 2.4 displays a part of the instance base with ten example instances. Each instance is stored as eight feature values, one class label (no instances map to more than one class for this task), and an occurrence count. The test instance **oreknew_** is matched against all 134 instances, and each match produces a distance. The minimal distance found during the matching process is 1, viz. with the instance **oreknow_** (from the word **foreknow**). As there is no other best-matching instance, the class of the best-matching instance, /-/, is taken as output. Thus, IB1 correctly produces /-/ as the classification of **oreknew_**: the **k** is not pronounced.

IB1-IG

IB1-IG (Daelemans and Van den Bosch, 1992a; Daelemans *et al.*, 1997a) differs from IB1 in the weighting function $W(f_i)$ (cf. Eq. 2.1). The weighting func-

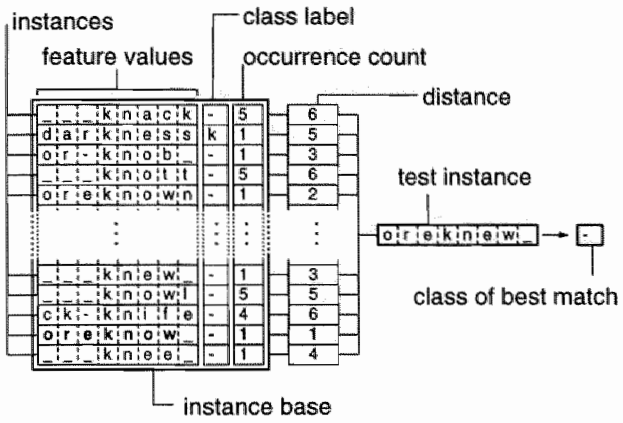


Figure 2.4: Visualisation of a part of the instance base constructed by IB1 applied to the **kn** task. The test instance **oreknew_** is matched against all stored instances. The class /-/ of the best-matching instance **oreknow_** (highlighted by bold-faced letters) is taken as classification of the test instance.

tion of IB1-IG, $W'(f_i)$, represents the *information gain* of feature f_i. Weighting features in k-NN classifiers such as IB1 is an active field of research (cf. Wettschereck, 1995; Wettschereck *et al.*, 1997, for comprehensive overviews and discussion). Information gain is a function from information theory also used in ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). Appendix C provides the details for computing the information gain of features given an instance base. The information gain of a feature expresses its relative relevance compared to the other features when performing the mapping from input to classification. Using the weighting function $W'(f_i)$ acknowledges the fact that for some tasks, some features are far more relevant (important) than other features. When $W'(f_i)$ replaces $W(f_i)$ in the distance function (Eq. 2.1), instances that match on a feature with a relatively high information gain are regarded as less distant (more alike) than instances that match on a feature with a lower information gain.

IB1-IG is a *filter model* (John, Kohavi, and Pfleger, 1994): the feature weights are computed before instances are classified. In contrast, in *wrapper models* (John *et al.*, 1994) optimal feature weights or selections are searched in a separate phase before testing, in which the induction algorithm is systematically trained and tested on subsets of the learning material with different feature

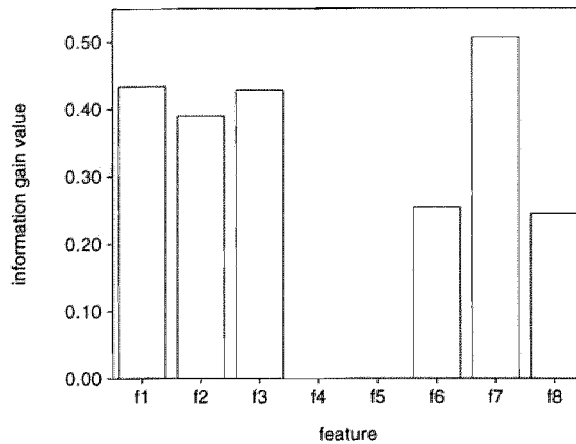


Figure 2.5: Information-gain values of the eight features of **kn** instances.

selections (John *et al.*, 1994). Searching for optimal feature selections (i.e., feature weights of 0.0 and 1.0) is computationally less costly than searching for optimal real-valued weights, which favours employing the wrapper approach in combination with feature selection (John *et al.*, 1994). However, Wettschereck, Aha, and Mohri (1997) note that real-valued weighting methods tend to outperform feature selection for tasks where some features are useful but less important than others. The latter tends to be the case for the (sub)tasks investigated here (cf. Appendix D), and it holds for the feature weights of the **kn** task. We choose to apply the filter approach because we suspect that searching real-valued feature weights in a wrapper approach is computationally very inefficient as compared to the filter-model approach, given the typical sizes of our data sets (cf. Section 2.3). Thus, our choice is motivated by considerations on feasibility; we do not exclude the possibility that a wrapper approach would yield similar or superior performance (e.g., in terms of generalisation accuracy), given enough computing resources and time.

Figure 2.5 displays the information-gain values computed over the instance base of 270 **kn** instances. As the fourth and fifth features have a single constant value (**k** and **n**), they have an information gain of 0. The feature with the highest information gain is the seventh feature, i.e., the second context letter to the right of **kn**. The feature with the second-highest information gain is the first feature (the third context letter to the left), closely followed by the third feature (the first context letter to the left).

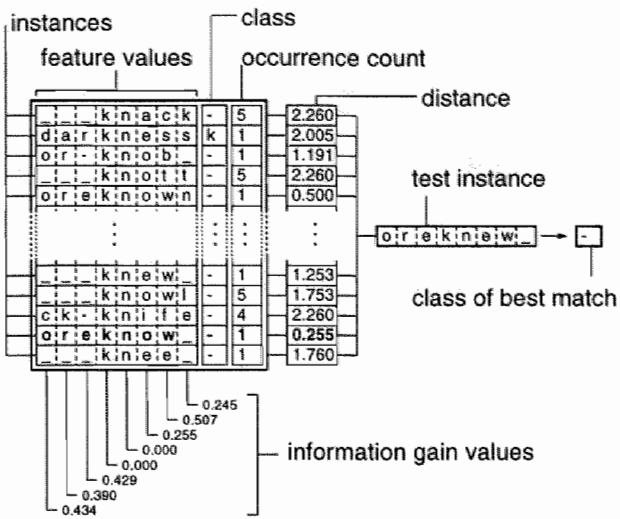


Figure 2.6: Visualisation of a part of the instance base constructed by IB1-IG applied to the **kn** task. An example test instance, **oreknew_**, is matched against all stored instances. The class /k/ of the best-matching instance (highlighted by bold-faced letters) is taken as classification of the test instance.

Applying IB1-IG to the **kn** task, an identical instance base is constructed as with IB1. The new information-gain-weighted similarity function now operates as exemplified in Figure 2.6, in which the test instance **oreknew_** is matched against ten example instances. IB1-IG finds one best-matching instance, viz. **oreknow_** (from **foreknow**), the same best-matching instance as found by IB1. Subsequently, IB1-IG correctly classifies **oreknew_** as /-/, i.e., the **k** is not pronounced.

2.1.4 Two non-pruning decision-tree learning algorithms

In this subsection we describe C4.5 (Quinlan, 1993), and IGTREE (Daelemans *et al.*, 1997a). Learning in C4.5 and in IGTREE takes the form of decision-tree learning, also known as top-down induction of decision trees (TDIDT). The decision-tree-learning approach is based on the assumption that similarities between subsets of instances can be exploited to compress an instance base as used by IB1 and IB1-IG into a decision tree, largely retaining the ability to generalise. Decision-tree learning is sometimes referred to as *eager learning*,

in contrast with the lazy learning of instance-based approaches, since a major computational effort of constructing decision trees occurs at learning time. Decision-tree learning is a well-developed field within AI. See, e.g., Safavian and Landgrebe (1991) for a survey; Quinlan (1993) for a synthesis of major research findings; and Hunt *et al.* (1966) and Breiman *et al.* (1984) for older accounts of decision trees from the statistical pattern-recognition area.

We begin by describing IGTREE, since it can be seen as an optimisation of IB1-IG, the previously described algorithm. It is a variation of C4.5, the description of which follows that of IGTREE.

IGTREE

IGTREE (Daelemans *et al.*, 1997a) was designed as an optimised approximation of IB1-IG. In IB1-IG, information gain is used as a weighting function in the similarity metric; in IGTREE, information gain is used as a guiding function to compress the instance base into a decision tree. Instances are stored in the tree as paths of connected nodes ending in leaves which contain classification information. Nodes are connected via arcs denoting feature values. Information gain is used in IGTREE to determine the order in which feature values are added as arcs to the tree[7].

Feature-value information is stored in the decision tree on arcs. The first feature values, stored as arcs connected to the tree's top node, are those representing the values of the feature with the highest information gain, followed at the second level of the tree by the values of the feature with the second-highest information gain, etc., until the classification information represented by a path is unambiguous. Knowing the value of the most important feature may already uniquely identify a classification, in which case the other feature values of that instance need not be stored in the tree. Alternatively, it may be necessary for disambiguation to store a long path in the tree. The difference in path lengths allows for a graded working definition of the concept of regularity: the more regular the instance is, the shorter its path in the tree.

The tree is compressed optionally in a second stage by recursively pruning all leaves which are labelled with the same class as their parent node, as the class information of these children does not contradict the (default) class

[7] The kind of tree built by IGTREE is related to the concept of *letter trie* proposed by Knuth (1973). A letter trie compresses a lexicon of letter strings (e.g., spelling words) into a trie structure, in which the letters of the strings are stored as (overlapping) arcs from left to right. Using letter tries, lexical look-up can be made more efficient, and less memory consuming.

information already present at the parent node. This compression does not affect IGTREE's classification accuracy.

Apart from storing uniquely identified class labels at each leaf, IGTREE stores at each non-terminal node information on the most probable classification given the path so far, according to the classification bookkeeping information maintained by the tree construction algorithm. The most probable classification is the most frequently occurring classification in the subset of instances being compressed in the tree. This information is essential when processing new instances. Processing a new instance involves traversing the tree by matching the feature values of the test instance with arcs the tree, in the order of the feature information gain. Traversal ends when (i) a leaf is reached or when (ii) matching a feature value with an arc fails. In case (i), the classification stored at the leaf is taken as output. In case (ii), we use the most probable classification on the last non-terminal node most recently visited instead.

For more details on IGTREE, see Appendix E, which provides a pseudo-code description of IGTREE's procedures for constructing decision trees and retrieving classifications from the trees; see also Daelemans *et al.* (1997a).

Applying IGTREE to the **kn** task, given the information-gain ordering of features as displayed in Figure 2.5, yields the decision tree as visualised in Figure 2.7. This tree contains thirteen nodes, of which seven are leaves containing unambiguous class labels. It can be seen from Figure 2.7 that at each level in the tree one feature is tested, starting with feature f_7, which is computed to have the highest information gain (see Figure 2.5), followed by f_1, and f_3, respectively; at the third level, all classifications are disambiguated.

The classification of the test instance **oreknew_** is retrieved from the tree in Figure 2.7 by a very short traversal: the value at the most important feature f_7 of **oreknew_**, viz. **w**, matches with none of the arcs stemming from the top node (viz. **c**, **g**, **m**, **s**, **t**, and **y**). When no matching feature value at an arc is found, traversal to the tree is stopped, and the classification of the most recently visited node is taken, which in this case is the top node. Since this node is labelled /-/ (i.e., the most frequently occurring class in the training set), /-/ is correctly taken as the classification of **oreknew_**. The **k** is not pronounced in **oreknew_**, according to IGTREE.

C4.5

C4.5 (Quinlan, 1993) is a decision-tree learning algorithm which basically

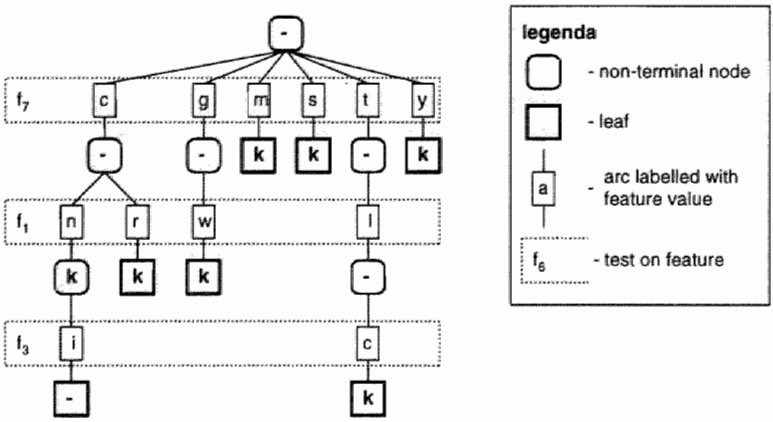


Figure 2.7: Visualisation of the decision tree after application of IGTREE on the **kn** task.

uses the same type of strategy as IGTREE to compress an instance base into a compact decision tree.

There are three relevant differences between both algorithms. First, C4.5 recomputes the information gain values of the remaining features in the subset of instances investigated for each new arc. In contrast, IGTREE computes the information gain of features only once on the basis of the complete training set.

Second, C4.5 does not implement the second compression stage of IGTREE. Instead it has an optional *pruning* stage, in which parts of the tree are removed as they are estimated to contribute to instance classification below a tunable *utility threshold*. This implies that in C4.5, the principle of storing all necessary knowledge for the classification of all training instances (as implemented in IGTREE) is not maintained when the utility threshold is set at less than 100%. Quinlan argues that this threshold should be set somewhat below 100% (e.g., at 90%) to prevent unwanted storage of low-occurrence exceptions or noise (Quinlan, 1993), usually referred to as small disjuncts (Holte *et al.*, 1989). Ignoring exceptions is what our inductive language learning approach aims to avoid (cf. Sections 1.2 and 2.1). Therefore, all experiments were performed with the utility threshold of 100%, i.e., pruning was disabled.

Third, C4.5 can consider *groups* of feature values as if they were single values. This is especially useful when it results in a higher information gain

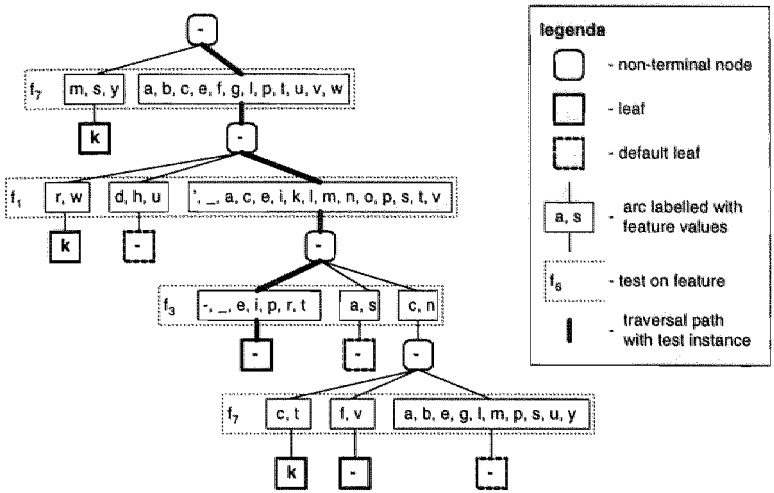


Figure 2.8: Visualisation of the decision tree after application of C4.5 on the **kn** task. The bold-faced line follows the retrieval of the classification of the **oreknew_** instance.

(ratio) (Quinlan, 1993). In the remainder of this thesis, C4.5 is applied with this parameter selected.

Figure 2.8 displays the tree generated by C4.5[8] when applied to the **kn** task. The tree contains twelve nodes. Retrieval of the classification of the test instance **oreknew_** from this tree proceeds as follows (and is displayed in Figure 2.8 by bold-faced lines): (i) the value at feature f_7 of **oreknew_**, **w**, matches with the value **w** in the second group of feature values distinguished by C4.5; (ii) a non-terminal node is met, consequently, traversal continues; (iii) the value at feature f_1, **o**, matches with the value **o** in the third group of feature values; (iv) a non-terminal node is met, consequently, traversal continues again; (v) the value at feature f_3, **e**, matches with the value **e** in the first group of feature values; (vi) a leaf node is met, labelled /-/: on meeting this leaf node traversal is halted and the class label /-/ is produced as (correct) output. According to C4.5, the **k** is not pronounced.

[8] Our experiments are performed with release 7 of C4.5.

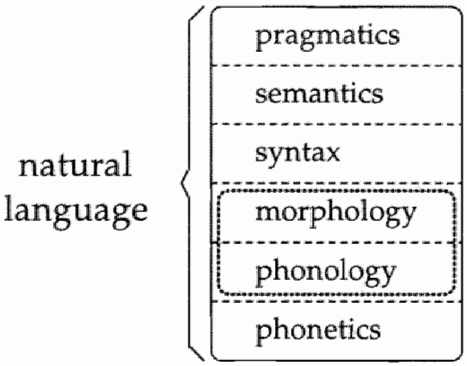


Figure 2.9: Classical decomposition of natural language into six linguistic domains, separated by dashed lines. The dotted-line box highlights the two domains investigated in this thesis, viz. morphology and phonology, in short morpho-phonology.

2.2 Theoretical background of word pronunciation

To provide a theoretical background on word pronunciation, we introduce the linguistic domains most relevant for word-pronunciation, viz. *morphology* (introduced in Subsection 2.2.1) and *phonology* (introduced in Subsection 2.2.2.

A classical linguistic view on natural language is the decomposition into six linguistic domains as displayed in Figure 2.9. It is believed that in human natural-language processing as well as in any system performing a natural-language task (e.g., producing utterances in a dialogue, or converting text to speech, Allen *et al.*, 1987), all components in Figure 2.9 play a role. This can be understood when language is seen as a mapping between thoughts to utterances, and vice versa. In both the generation and analysis of language, knowledge of how a person speaks or writes is intricately coupled to that person's knowledge of the world.

Our focus is on a language process which takes written words as input, and produces phonemic transcriptions as output. This conversion of spelling words to phonemic transcriptions is referred to as word pronunciation. In the process of word pronunciation, phonological processing is strongly interrelated with morphological processing. Word pronunciation is a process occurring both in the morphological and in the phonological domain. Expressing this strong relationship between the two domains with respect to

word pronunciation, we refer to the combination of both domains with the term morpho-phonology.

2.2.1 Morphology

Morphology is the language-specific (or speaker-specific) knowledge about the internal structure of words: it describes the building blocks, or *morphemes*, of which words are made up (cf. Matthews, 1974; Bauer, 1988). Although in general we may state that morphological structure of words conveys information (Sproat, 1992), large differences exist between languages in the kind of information it encodes. Extremes are on the one hand *isolating* languages in which almost no words contain more than one morpheme, such as Mandarin Chinese, and on the other hand *agglutinating* languages in which single words can contain long sequences of morphemes carrying both syntactic and semantic knowledge, such as Turkish and Finnish (Bloomfield, 1933), and *incorporating* languages, such as Inuktitut (an Inuit language), in which sentences contain only one word which is made up of strings of morphemes.

Two types of morphemes are generally distinguished: *free* morphemes and *bound* morphemes. First, free morphemes can occur in isolation. Consequently, in ideal isolated languages all morphemes are free. In English, examples of free morphemes are **dog**, **safe**, and **walk**. Second, bound morphemes must be attached to other morphemes and cannot occur in isolation. Examples of bound morphemes in English are **al** at the end of **magical**, **re** at the beginning of **restart**, and **s** at the end of **dogs**.

There are two ways in which morphemes can attach. First, *derivation*, or *word formation* (Sproat, 1992), is the kind of attachment which takes as input a word (which may already be morphologically complex) and a *derivational* morpheme (which can be free or bound), and produces as output a different word *derived* from the original word. Second, *inflection* is the kind of attachment which takes as input a word and an *inflectional* morpheme (which is always bound), and produces as output the same word, in a form appropriate to the particular context. An important difference between derivation and inflection that follows from these definitions is that in English, derivation can lead to the formation of a word of a different syntactic word class (e.g., **grammatical** is an adjective; attaching the derivational morpheme **ity** leads to **grammaticality**, which is a noun). As a rule, inflection never leads to a word of a different syntactic word class (attaching **s** to the verb **read**, gives **reads**, which is still a verb).

As we are focusing on English word pronunciation, a closer look at the particular complexity of English morphology is appropriate (cf. Bauer, 1983). First, English has a rather productive morphology, in the sense that morphologically complex words such as **antidisestablishmentarianism** can be formed fairly freely. Finding all morphemes in a word thus involves knowing all morphemes in English, and being able to disambiguate between different possible segmentations (e.g., **booking** is most probably composed of the morphemes **book** and **ing**, and not of **boo** and **king**). English morphology is not as productive as Dutch or German morphology, however, because derivation by attaching two or more free morphemes (i.e., *compounding* as in **football**, **lighthouse**, **handwriting**) occurs relatively infrequently (Sproat, 1992): the productivity of English morphology is mostly limited to the attachment of bound morphemes and inflections to words; moreover, the number of bound morphemes and inflections in English is limited.

Second, the attachment of morphemes in English often leads to spelling changes of either one of the morphemes, which makes recognition of these morphemes in a word more difficult. Four types of spelling changes occur: (i) etymological spelling changes caused by *co-articulatory* effects in the pronunciation of a word (e.g., **impossible** is composed of **in** and **possible**; the **n** changed into **m** because the **n** is pronounced /m/ when preceding a /p/); (ii) the replacement of letters by other letter groups (e.g., the **y** in **ty** is replaced by **ie** with plural inflection: **entity** plus **s** becomes **entities**); (iii) the insertion of letters (e.g., an **e** is inserted in the plural inflection of **church** plus **s**, causing **churches**), and (iv) the deletion of letters (e.g., an **e** is deleted in the past tense inflection of **love** plus **ed**, causing **loved**).

Third, English has a fairly regular system of *morphotactics*, i.e., the constraints governing the order of morphemes in a word. Sproat (1992) presents the example of **motorizability**, which is the only order in which the four morphemes **motor**, **ize**, **able** and **ity** can occur: **itymotorizable**, **izemotorability**, **motorableizity**, and **motorabilityize** are examples or orderings that will never occur in English.

Most of English morphotactics can be represented by *finite-state automata* (FSAs) (Koskenniemi, 1984; Sproat, 1992). Given these morphotactics and given lexical knowledge of all morphemes in English, a good deal of English words can be morphologically analysed, and morpheme boundaries can be found within words. However, an additional mechanism is needed recognising possible spelling changes near morpheme boundaries. It has been argued (Koskenniemi, 1984) that in order to recognise such spelling changes, *two-level*

finite-state transducers (FSTs) are needed. An FST operates on two representations, stored on *tapes*, of the word to be analysed, rather than one. One tape is called the *surface tape*, storing the representation of the word as it actually appears, e.g., **churches**. The second tape is called the *lexical tape*, containing the *deep* structure of the word, i.e., a concatenation of the morphemes it is composed of, e.g., **church+s**. Reacting to the claim that relatively complex FSTs are mandatory for morphological analysis, Sproat (1992) calls for future work on models less dominated by multiple-level finite-state morphotactics, as their complexity is unfavourable, unless parallel machines could be used (Koskenniemi and Church, 1988). Our investigations of learning morphological segmentation focus on single-level processing, which is finite-state but does not have, in principle, the computational disadvantages as multi-level finite-state morphotactics.

2.2.2 Phonology

The phonology of a language is the collection of knowledge of the *sound patterns* occurring in spoken language, be it speaker-specific (De Saussure's (1916) *parole*), language-specific (De Saussure's (1916) *langue*), or language-universal (Kenstowicz, 1993; Goldsmith, 1996). There is no single theory of phonology; it is studied in linguistics (Chomsky and Halle, 1968; Kenstowicz, 1993; Mohanan, 1986; Goldsmith, 1996) and in psycholinguistics (Levelt, 1989). In linguistics, one influential area of the study of phonology has adopted the Chomskyan approach to language, transferred to phonology in the form of *generative phonology* (Kenstowicz, 1993). *Lexical phonology* (Mohanan, 1986; Kenstowicz, 1993) is an extension of generative phonology in which the interaction between phonology and morphology is explicitly modelled. Inherent, undesired complexity of certain assumptions in generative phonology led to the development of two theories of non-linear phonological processing: autosegmental phonology or two-level phonology (Goldsmith, 1976; Bird, 1996) and metrical phonology (Liberman and Prince, 1977). In autosegmental phonology, the single phonemic representation is expanded by introducing a second *autosegmental* level, in which certain abstractions over the phonemic representation can be represented and manipulated independent of the phonemic representation. In metrical phonology, tree structures or grids are used to represent hierarchical structures on top of the one-dimensional phonemic representations. In psycholinguistics, many pluriform models of phonology and phonology-related phenomena such as reading aloud have been proposed (e.g., Coltheart, 1978; Glushko, 1979; Levelt, 1989),

displaying a lack of consensus in the psycholinguistic field on a single model of phonology and its role in generating and understanding utterances.

From the multitude of phonological theories, it follows that developers of word-pronunciation systems cannot adopt a single theory straightforwardly, and have to choose between alternatives. Practice shows that implemented word-pronunciation systems often combine fragments of morphological and phonological theories for building components of the system, augmented with heuristics, preprocessing, and postprocessing procedures (Allen *et al.*, 1987). When a fragment of a theory describes an isolated phenomenon, this fragment or subtheory can be said to describe a phonological abstraction level. If defined precisely enough, the abstraction level can be implemented as an isolated component of the word-pronunciation system. Such components are usually referred to as *modules* (a classic example of such a modular system is the MITALK system. Allen *et al.*, 1987). Thus, from the viewpoint of building word-pronunciation systems, the issue with respect to phonological theory is which phonological abstraction levels may provide good bases for building modules. In lexical phonology, explicit proposals are made for abstraction levels between underived phonological words (entries in a lexicon) and actual phonological transcriptions of words (Kenstowicz, 1993), but none of these levels refers to spelling – in general, phonological theory does not concern itself with the level of spelling. Rather, spelling is referred to the study of *graphematics* (i.e., the study of writing systems, Venezky, 1970; Sampson, 1984; Coulmas, 1989; Röhr, 1994). Alternatively, in psycholinguistic models, abstraction levels are assumed in order to account for empirical findings (e.g., statistical features of text, speech, and speech errors, cf. Levelt, 1989, or word naming reaction times, cf. Coltheart, 1978; Jared, McRae, and Seidenberg, 1990; Plaut, McClelland, Seidenberg, and Patterson, 1996). Rather than making a comprehensive inventory of abstraction levels assumed in various theories, we restrict ourselves to describing the following four levels which can be found commonly in word-pronunciation systems (Hunnicutt, 1980; Allen *et al.*, 1987; Daelemans, 1988; Daelemans, 1987; Coker *et al.*, 1990):

(i) the *graphemic level*, in which words are represented as strings of graphemes;

(ii) the *phonemic level*, in which words are represented as strings of phonemes;

(iii) the *syllabic level*, in which words are represented as strings of syllables; and

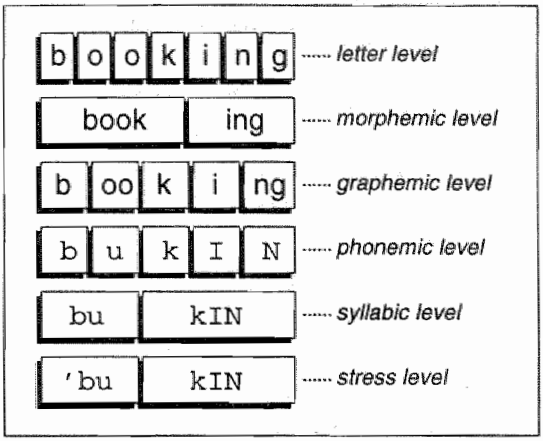


Figure 2.10: Illustration of the letter level, the morphological level, and the four phonological levels, using the example word **booking**. At each level, the word is represented as a string of elements specific for the level.

(iv) the *stress level*, in which words are represented as strings of stressed syllables.

In sum, we will henceforth assume the following six levels: the letter level (which is given and does not have to be analysed, unless of course by optical character recognition), the morphemic level described in Subsection 2.2.1, and the four phonological levels. We will employ these six levels as levels of abstraction in the word-pronunciation systems described in Chapters 3 to 6. Figure 2.10 illustrates all levels, using the example word **booking** to illustrate which elements are used in each level to represent the word.

The graphemic level

On the graphemic level, a word is represented as a string of *graphemes*. A grapheme is a letter or letter group realised in pronunciation as one phoneme. The graphemic and phonemic levels are therefore strongly related. The task of finding the graphemes in a word is referred to as *graphemic parsing* (Van den Bosch *et al.*, 1995). In English, graphemes can contain up to four letters, e.g., **ough** in **through**. The example word **booking**, as displayed in Figure 2.10, contains five graphemes: **b**, **oo**, **k**, **i**, and **ng**. The grapheme is not directly covered by any phonological theory; rather, it is proposed as a language

element in studies of writing systems (e.g., Röhr, 1994) and graphematics (Venezky, 1970), and as an element at the abstraction level between letters and phonemes in word-pronunciation systems (Allen *et al.*, 1987). Wolters (1997) argues that the underrated role of the grapheme in phonological theories can be attributed to the primacy of the phoneme as the main element of utterances, as advocated by De Saussure (1916).

There are no straightforward, unambiguous (one-to-one) mappings between letters and phonemes in English (nor are there in, e.g., Dutch or French, Van den Bosch *et al.*, 1995); rather, the relations are generally many-to-many (Venezky, 1970). This is in contrast with the straightforward writing systems of, e.g., Czech and Serbo-Croatian, in which relations between letters and phonemes are for the major part one-to-one: such writing systems are termed *shallow orthographies* (Katz and Frost, 1992). In English, however, which has a *deep orthography* (Katz and Frost, 1992), one grapheme can be associated to many phonemes (e.g., **ea** can be pronounced as /i/, /ɛ/, /ɑː/, and /ɪə/), and (ii) several graphemes can be associated to the same phoneme (e.g., **f**, **ff**, **ph**, and **gh** can all map to /f/). Virtually all of these mappings are context-dependent, and sometimes occur only once (e.g., the phonemic mapping of **gh** is /p/ only in the word **hiccough**, which is usually spelled **hiccup** nowadays).

Apart from its strong relation to the phonemic level, the graphemic level is related to the morphological level as well, in the sense that graphemic parsing needs information from the morphological level in order to operate correctly. For example, one needs to know that a morpheme boundary occurs between the **p** and **h** of **loophole**, to prevent the incorrect parsing of the two-letter grapheme **ph** rather than the correct parsing of the two single-letter graphemes **p** and **h**. Morphology thus obfuscates the equivalence in similarity at the levels of writing and pronunciation in English: it is an important cause for $S \approx S'$ rather than $S = S'$ (cf. Section 1.2). Two other causes for $S \approx S$ rather than $S = S'$ are the different paces of change in pronunciation and in spelling, and the incorporation of strange spellings by loaning words and their pronunciations from other languages (Röhr, 1994; Coulmas, 1989).

The phonemic level

On the phonemic level, words are represented as strings of *phonemes*. Phonemes are abstract mental representations of *phones* (Kenstowicz, 1993). Phones are sound patterns in speech, uniquely characterised by articulatory feature settings determining the physics of the vocal organs during pronunciations of the phones (Clements and Hume, 1995). For example, the phone

[p] can be uniquely characterised by the presence of the articulatory features *labial* (i.e., involving the touching of both lips), *stop* (i.e., involving a temporary halt of the air flow through the mouth, followed by a plosive release of the air), and *unvoiced* (i.e., pronounced without the vocal folds vibrating). No other phone is characterised by these articulatory features.

While phones are physically grounded, phonemes are abstract mental representations of phones (Kenstowicz, 1993). An example in English of a phoneme is the phoneme /t/. Although /t/ may be realised as different phones, as in **stem**, **ten**, **atom**, **bottle**, or **pants**, it is generally perceived as the same sound /t/. A distinguishing attribute of phonemes, not possessed by phones, is that when replacing one phoneme in a word by another phoneme that may differ in only one articulatory feature, the word may become another word. For example, changing the /p/ in /pæt/ (**pat**) to /b/ renders /bæt/, which is the phonemic transcription of a different word, **bat**.

The example word **booking** is represented at the phonemic level by five phonemes, each corresponding to each of the five graphemes in **booking** (in brackets): /b/ (**b**), /u/ (**oo**), /k/ (**k**), /ɪ/ (**i**), and /ŋ/ (**ng**).

The cooccurrence of phonemes in English words is regulated by the *phonotactics* of English (Chomsky and Halle, 1968; Goldsmith, 1995). These phonotactics cannot be described on the phonemic level, since an additional element is needed to describe them according to present-day mainstream phonological theories, viz. the syllable (Kenstowicz, 1993).

The syllabic level

The role of the syllable in phonology has been controversial in phonological theory (Kenstowicz, 1993). Chomskian phonology, described in SPE (Chomsky and Halle, 1968), did not consider the syllabic level at all. On the syllabic level, a word is represented as a string of *syllables*. The syllable is in its turn a string of one or more phonemes. A syllable consists of three parts: (i) an *onset* consisting of zero to three consonants, (ii) a *nucleus* consisting of a *monophtong* (single vowel) or *diphtong* (double vowels such as /ʊə/ as in **poor**), and (iii) a *coda* again consisting of zero to three consonants (Selkirk, 1984; Blevins, 1995).

Within the context of the syllable, the phonotactics of English strongly constrain the phonemes that can occur as neighbours. For example, the onset /rt/ is impossible in English; /tr/ is perfectly acceptable (as in **track**); alternatively, /tr/ cannot occur as a coda, whereas /rt/ does occur in English (as in **art**). These particular phonotactical constraints within the syllable are referred to as the *sonority sequencing principle* (Selkirk, 1984; Blevins, 1995).

Syllable boundaries in polysyllabic words are determined by the *maximal onset principle* (MOP) (Treiman and Zukowski, 1990), which states that between two nuclei, as many consonants belong to the second phoneme (seen from left to right) as can be pronounced together. For example, /bukɪŋ/ (**booking**) is syllabified as /bu-kɪŋ/; /ɛntrɪ/ (**entry**) is syllabified as /ɛn-trɪ/, since /tr/ is a well-formed onset, but not /ntr/.

The stress level

On the stress level, a word is represented as a string of syllables with *stress markers*. Stress markers denote the realisation of a certain prominence of one of the syllables in a word. All words have one primary stress position; polysyllabic and polymorphemic words sometimes have a less prominent, secondary stress position (e.g., as in the pronunciation of **energetically**, /ɛ-nə-ʤɛ-tɪ-kə-lɪ/, in which the third syllable /ʤɛ/ receives primary stress, and the first syllable /ɛ/ receives secondary stress). For an detailed overview of language-universal phonological theories on word stress, cf. Kager (1995).

In English, the placement of word stress is highly irregular. There is a weak tendency for content words (i.e., nouns, adjectives, verbs) to have primary stress on the first syllable (as is the case for the example word **booking**, which receives primary stress on the first syllable) (Cutler and Norris, 1988). Word stress is furthermore strongly related to the morphological structure of words: consider, for example, the difference in word stress in **photograph** and **photography**, caused by the attachment of the stress-affecting bound morpheme **y**; the difference in stress of **ball** in **ballroom** and **football**; the difference in stress of **anti** in **antipathy** and **antipathetic**. It suffices here to state that determining the place of primary and secondary stress in polysyllabic English words is hard; in polymorphemic English words, it is also mandatory to know the morphological structure (Chomsky and Halle, 1968; Allen *et al.*, 1987).

2.3 Word-pronunciation data acquisition

The set-up of our study demands that we acquire instances of the word-pronunciation task in order to perform experiments with the selected learning algorithms (cf. the third requirement formulated in Section 2.1). We describe the acquisition of word-pronunciation instances in Subsection 2.3.

The resource of word-pronunciation instances used in our experiments is the CELEX lexical data base of English (Van der Wouden, 1990; Burnage, 1990)

information type	field #	CELEX code	contents
orthography	1	WordSylDia	Hyphenated orthographical representation of the word
phonology	2	PhonStrsDISC	Syllabified phonemic transcription with stress markers
morphology	3	FlectType	Type of inflection
	4	TransInfl	Transformations caused by inflection
	5	MorphStatus	Morphological status
	6	MorphCnt	Number of derivational analyses
	7	StrucLabLemma	Hierarchical, bracketed, labelled derivational analyses

Table 2.2: The seven information fields extracted from CELEX.

(used earlier with the **kn** task, cf. Subsection 2.1.1). All items in the CELEX data bases contain hyphenated spelling, syllabified and stressed phonemic transcriptions, and detailed morphological analyses. We extracted from the English data base of CELEX this information, restricting the abundance of analyses and numbers provided by CELEX to the seven information fields and values displayed in Table 2.2. The resulting data base contains 77,565 items. Table 2.3 provides some randomly selected examples of items contained in our English data base. The numbers in the columns of Table 2.3 correspond to the numbers of the information fields listed in Table 2.2.

Spelling words and their phonemic counterparts are composed of 42 different letters (including letters with diacritics such as **é** and **ö**) and 62 different phonemes, respectively. Appendix A lists both alphabets. Both sets of letters and phonemes are adapted from the ones used in CELEX, with two minor additions directly related to our experimental set-up: (i) the space before and after a word is also counted as a member of the letter alphabet, and is denoted by _ (cf. Subsection 2.1.1), and (ii) eight double phonemes represented by single characters are added to the phonemic alphabet to ensure that no phonemic transcription contains more phonemes than its spelling counterpart contains letters (cf. Sections 3.2 and 3.3).

All items in the 77,565-word data base are unique; most items, however, share some information fields with other similar items. For example, verb inflections derived from the same verb stem share the same derivational analysis. In Chapter 3, in which isolated morpho-phonological subtasks are

1	2	3	4	5	6	
dark-ness	'd#k-n@s	S		C	1	...
king-li-est	'kIN-lI-Ist	s	@-y+iest	C	1	...
left--wing-ers	"lEft-'wI-N@z	P	@+s	C	2	...
sight	's2t	S		M	1	...
val-i-dat-ing	'v{-lI-dI-tIN	pe	@-e+ing	C	1	...

	7
...	((dark)[A],(ness)[N|A.])[N]
...	((king)[N],(ly)[A|N.])[A]
...	(((left)[A],(wing)[N])[N],(er)[N|N.])[N] ((left)[A],((wing)[N],(er)[N|N.])[N])[N]
...	(sight)[N]
...	((valid)[A],(ate)[V|A.])[V]

Table 2.3: Sample items in the English morpho-phonological data base.

investigated, additional descriptions will be given of the specific training and test sets extracted from the original data base.

It is noteworthy that CELEX includes both American and British spelling, i.e., it contains, e.g., **monolog** and **monologue**, and **generalize** and **generalise**. Hyphenation in CELEX roughly follows the principles of British English, as opposed to the more liberal hyphenation rules of American English. The phonemic transcriptions extracted from CELEX are the primary pronunciations as laid down in the *English Pronouncing Dictionary* (Jones *et al.*, 1991), which is aimed at representing the so-called 'Received Pronunciation' of British English.

The information contained in CELEX can only to a certain extent be seen as raw material. A large amount of linguistic assumptions underly the representations in CELEX, notably the morphological representations. We should therefore be aware of the fact that directly extracting information from CELEX implies copying some of the linguistic bias in the CELEX data. When this is the case, we define the subtask in such a way that this bias is maximally limited, also when this means limiting the granularity of information provided by CELEX.

2.4 Methodology

We compare the generalisation abilities of inductive-learning algorithms applied to the word-pronunciation task and morpho-phonological subtasks, by training and testing them on identical training and testing material. We exemplified our approach with a simple example task, the pronunciation of **kn**. In this section, we formalise our experimental methodology for inductive language learning in order to show that the fourth requirement for our investigation of inductive learning of word pronunciation is met. For each task or subtask T, we perform the following procedure:

1. From a real-world resource R_T containing instances of T, we extract an instance data base D_T. D_T contains as many instances as can be sampled from R_T.

2. We train (a subset of) the learning algorithms on a subset of D_T, the training set D_T^{train}, resulting in induced systems for performing T. All algorithms are trained on the same material.

3. We perform a fixed number of experiments by which we estimate for each induced system its *generalisation error*, i.e., the percentage error of incorrectly processed *test instances*. This is done by presenting all systems with an identical set of instances, the subset of D_T that is the complement of D_T^{train}, viz. the test set D_T^{test}.

4. We compare the generalisation errors of all induced systems and analyse their differences with statistical significance tests.

The extraction of instances from items in CELEX is described in Subsection 2.4.1. We provide details on how we conduct experiments applying learning algorithms to training and test sets of (sub)tasks in Subsection 2.4.2. The parameter settings of the algorithms used throughout the thesis are listed in in Subsection 2.4.3.

2.4.1 Windowing reformulated

As was illustrated with the **kn** task, the raw language data has to be formatted in such a way that the instances presented to the learning algorithms have a fixed size. A representation of a (sub)task must be determined in which the input is represented as feature-value vectors of a fixed length n. For example,

instances of the **kn** task contained eight feature values (cf. Subsection 2.1.1). In contrast with the **kn** task, the (sub)tasks investigated in the subsequent chapters are not defined as finding correct classifications of a particular letter, but as finding correct classifications of all letters in a word. Thus, words are not treated as single instances, but as series of instances. For each word, the number of instances equals the number of letters in the word. Therefore, the windowing method is reformulated as follows:

- it generates as many feature-value vectors per word as the number of letters in the word, by taking each letter separately as the middle letter in the window; and
- it generates feature-value vectors of length 7, viz. three left-context neighbour letter positions, one focus letter position, and three right-context neighbour letter positions.

For the remainder of this thesis, the feature-value vector length is fixed to $n = 7$. The locality assumption, formulated earlier with the **kn** task, is thus maintained. The fixed contextual scope is a limitation for certain (sub)tasks, as it does not include the long-distance scope needed in certain exceptional cases. We will critically discuss the consequences of fixing the contextual scope to seven neighbour letter positions in Chapter 7.

Figure 2.11 illustrates the windowing method applied to the example word **booking**. At each snapshot, the window captures one of the seven letters of the word, and includes three left and three right neighbour letters.

2.4.2 Cross-validation and significance testing

When we apply learning algorithms to morpho-phonological (sub)tasks, we are not primarily concerned with the question whether these learning algorithms can learn and reproduce the training material, since keeping all training material in memory would suffice in that case. Rather, our analyses focus on the ability of the learning algorithms to use the knowledge accumulated during learning for the classification of new, unseen instances of the same (sub)task, i.e., we measure their generalisation accuracy. To this end we have to establish a sound experimental methodology: techniques are available which meet this soundness requirement. Weiss and Kulikowski (1991) describe *n-fold cross validation* (n-fold CV) as a procedure of three steps. (i) On the basis of a data set, n partitionings are generated of the data set into one training set containing $((n-1)/n)$th of the data set, and one test set containing

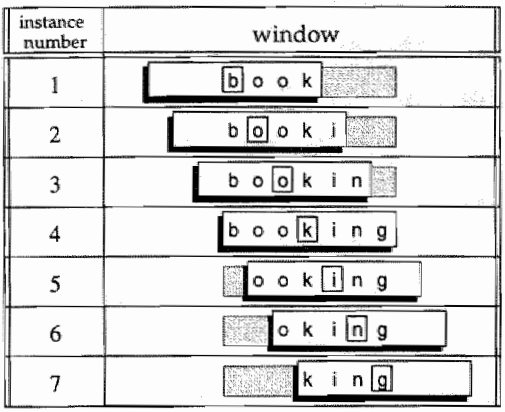


Figure 2.11: Example of the windowing method applied to the word **booking**. The fixed-size window (large rectangular box) captures a focus letter (small middle box) surrounded by three left and three right neighbour letters.

$(1/n)$th of the data set, per partitioning. (ii) For each nth partitioning, a learning algorithm is applied to the training set, and the generalisation error[9] is collected by applying the learned model to the test set. (iii) A mean generalisation error of the learned model is computed by averaging the generalisation errors found during the n experiments. Weiss and Kulikowski (1991) argue that by using n-fold CV, preferably with $n \geq 10$, one can retrieve a good estimate of the true generalisation error of a learning algorithm given an instance data base.

To make claims with respect to the relation of the n outcomes of an n-fold CV experiment to other experimental outcomes (e.g., of other learning algorithms applied to the same data set using an n-fold CV set-up), statistical analyses are available on the basis of which one can claim that, e.g., the generalisation accuracy of a learning algorithm is *significantly* better than that of another algorithm. All experiments reported in this thesis use a 10-fold CV set-up, combined with one-tailed t-tests (Hays, 1988). Given two 10-fold CV generalisation accuracies of two algorithms A and B, the probability that the generalisation accuracy of A is indeed better than that of B is 0.95 if and only if a one-tailed t-test returns a p-value $p < 0.05$. When we report on

[9] Henceforth, we also refer to *generalisation error*, being the reverse of *generalisation accuracy*; low generalisation error is high generalisation accuracy.

significances, we provide both the t-value (Hays, 1988) and the range of the p-value: $p \geq 0.05$ (i.e., the difference is not significant), $p < 0.05$, $p < 0.01$, or $p < 0.001$ (i.e., the difference is significant).

From Chapter 3 onwards, specific applications of learning algorithms to morpho-phonological (sub)tasks are sometimes analysed with additional methods related to the task at hand; these methods are introduced in the appropriate (sub)sections.

Salzberg (1995) provides a thorough critique on common practice in machine-learning research concerning the comparison of algorithms under certain experimental conditions. He distinguishes between comparisons of learning algorithms applied to benchmark tasks (e.g., contained in the PROBEN1 (Prechelt, 1994) or UCI (Murphy and Aha, 1995) benchmark collections), and comparisons of learning algorithms applied to real-world tasks. Our work belongs to the second type. Salzberg argues that for real-world tasks, straightforward comparisons of algorithmic generalisation accuracies (i.e., comparisons of the outcomes of two 10-fold CV experiments using one-tailed t-tests) may suffice, but any conclusions from such comparisons will be limited to the domain under investigation.

Apart from running 10-fold CV experiments to be able to compare accuracies of different algorithms, we also compute a *baseline accuracy* with each subtask under investigation. We introduce DC, which optionally stands for *Default Classifier* or *Dumb Classifier*. DC is basically an instance-based learning algorithm in which the learning component is identical to that of IB1 and IB1-IG, i.e., plain storage of all training instances in an instance base, and in which the classification component performs the following two simple ordered rules: given a test instance X,

1. match X against all stored instances in the instance base; as soon as an identical memory instance Y is found (i.e., all feature values of X and Y are equal), produce the (most frequently occurring) classification associated with Y as the classification of X;

2. if no identical instance to X exists in the instance base, produce the most frequently occurring classification in the whole instance base as the classification of X.

The combination of measuring the *overlap* between training and test sets (rule 1) and employing the *class bias* in the training set (rule 2) as a fall-back option, is taken here as the baseline score reported with the experiments in this thesis. With linguistic tasks, such as the **kn** task investigated in Section 2.1, using the overlap and bias can provide relatively accurate guesses.

For example, 126 instances in the **kn** data base occur more than once, e.g., the **__knigh** instance occurs eleven times (this is because the windowing method produces *parts* of words that are identical, while the words themselves are all unique). When one of these eleven **__knigh** instances is in the instance base, and the remaining ten instances are in the test set, these instances can all be classified correctly by DC because of the single memorised instance. As for the bias of the training material, we noted that 82.8% of all cases mapped to class /-/, which makes it a good guess for new, unseen instances (cf. Yvon, 1996 for similar findings). In the two-dimensional space defined by the learning effort and classification effort dimensions, visualised in Figure 2.1, DC can be placed in the lower-left corner.

2.4.3 Parameter setting

To optimise comparability of our experiments, we do not determine a new set of parameter settings for each application of a learning algorithm. Instead, we determine beforehand a number of default parameter settings for each algorithm which are kept constant throughout all experiments. Salzberg (1995) expresses some concerns on setting parameters. He argues that when using the same data for training and testing, any parameter adjustment during any experiment heavily decreases the significance levels that should be obtained to reach significant differences (Salzberg, 1995). He explicitly states that any parameter tweaking should be performed *before* test material is used by the experimenter. Following Salzberg's (1995) suggestion, we have adjusted the parameters of the five algorithms only once before the introduction of test material in the experiments.

The parameter settings of the learning algorithms used in our experiments are displayed in Table 2.4. For IB1, IB1-IG, and IGTREE, these parameter settings are known from past research (Daelemans and Van den Bosch, 1992a; Daelemans *et al.*, 1997a) to be typical or default for the specific algorithm. To determine the learning rate, the momentum, and the number of hidden units for BP, we performed a limited number of pilot experiments on a data set representing the task of English word hyphenation (Van den Bosch *et al.*, 1995) monitoring the error on training material under different parameter settings, and selecting the parameter settings that led to the highest accuracy on the classification of training material.

It can be expected that the fixed settings in Table 2.4 are not optimal for every experiment performed. Any possible disadvantage from fixing parameters at non-optimal settings apply especially to BP (seven parameters)

algorithm	parameter	setting
BP	learning rate	0.1
(Rumelhart *et al.*, 1986)	momentum	0.4
	# hidden layers	1
	# hidden units	50
	update tolerance	0.2
	patience threshold	0.025%
	halting patience	2
IB1	similarity matching	1-nn
(Aha *et al.*, 1991)		
IB1-IG	similarity matching	1-nn
(Daelemans *et al.*, 1997a)	feature weighting	information gain
IGTREE	feature ordering	information gain
(Daelemans *et al.*, 1997a)		
C4.5	feature ordering	gain ratio
(Quinlan, 1993)	number of trees built	1
	subtree pruning	no
	feature-value grouping	yes

Table 2.4: Fixed parameter settings of learning algorithms.

and to C4.5 (four parameters); the other three algorithms function under one or two parameters. Although we acknowledge the potential disadvantage for fixing parameter settings, especially for the algorithms with many parameters, BP and C4.5, our purpose here is to compare a set of learning algorithms under identical experimental conditions (i.e., using 10-fold CV and fixed parameters) applied to a set of morpho-phonological (sub)tasks, instead of minimising the generalisation error of any specific algorithms applied to any specific (sub)tasks.

Chapter 3

Learning word-pronunciation subtasks

In Section 2.2 we described six levels of abstraction assumed by developers of mainstream word-pronunciation systems. The assumption is, inspired by theories of morphology and phonology, that performing word pronunciation implies performing a stepwise transcription from the level of letters to the level of stressed phonemes by performing the subtasks in sequence. Leaving aside the level of letters, which is the input to word pronunciation that is given and does not need to be computed, brief analyses of the remaining five abstraction levels (viz. the morphological, graphemic, phonemic, syllabic, and stress levels) indicated that each level represents a complex subtask in itself. It is therefore a relevant question whether the selected algorithms can learn these word-pronunciation subtasks and attain adequate generalisation accuracy. If one adopts the mainstream-linguistic argument that the abstraction levels are necessary, low generalisation accuracies on any of the subtasks would reveal limitations of inductive learning that probably would also apply to learning the word-pronunciation task as a whole. Alternatively, a successful learning of all subtasks with high generalisation accuracies would indicate that successful inductive learning of the word-pronunciation task might be possible.

In this chapter we investigate in isolation the application of the five inductive-learning algorithms to each of five subtasks related to the five abstraction levels. We have argued earlier (cf. Section 2.2.2, page 46) that these five abstraction levels are relatively common in morphological and phonological theories (Kenstowicz, 1993; Levelt, 1989; Goldsmith, 1996); they also

occur implemented in modules in existing text-to-speech synthesis systems (e.g., Allen *et al.*, 1987; Daelemans, 1988; Coker *et al.*, 1990). The five subtasks investigated in this chapter are employed in the subsequent chapters, from Chapter 4 on, as components in word-pronunciation systems.

Our approach to this investigation is basically analogous to the treatment of the **kn** task in Section 2.1. The five subtasks are the following (between brackets we mention the section in which the application of learning algorithms to the subtask is reported):

1. morphological segmentation (Section 3.1),
2. graphemic parsing (Section 3.2),
3. grapheme-phoneme conversion (Section 3.3),
4. syllabification (Section 3.4), and
5. stress assignment (Section 3.5).

Subtask 1, morphological segmentation, is to detect morpheme boundaries within words (cf. Subsection 2.2.1). Subtasks 2, 3, 4, and 5 have as output the four elements described in Subsection 2.2.2, viz. graphemes, phonemes, syllables, and stress markers. Subtask 2, graphemic parsing, is the detection of graphemes in spelling words. Subtask 3 is the conversion of graphemes to phonemes. Subtask 4, syllabification, is the detection of syllable boundaries in phonemic transcriptions. Finally, subtask 5, stress assignment, is the placement of stress markers on phonemic transcriptions.

The five subtasks are investigated *in isolation*, viz. they are defined according to the following three rules:

(i) For each subtask we create the instances directly from our English data base (as with the **kn** task, cf. Subsection 2.1.1).

(ii) In the case of subtasks 1, 2, and 3, the feature values represent letters, and in the case of subtasks 4 and 5, the feature values represent phonemes. No other information (e.g., morpheme boundary markers, graphemic parsings, syllable boundary markers, or stress markers) is included in the instance features.

(iii) The possible classifications are strictly the classifications of the subtask itself. For example, the possible classifications of the grapheme-phoneme-conversion subtask are phonemes, without additional classes such as stress markers or syllable boundaries.

After the presentation of the experimental results with the five subtasks in Sections 3.1 to 3.5, the results are summarised and evaluated in Section 3.6.

instance number	left context			focus letter	right context			classification
1	_	_	_	**b**	**o**	**o**	**k**	1
2	_	_	**b**	**o**	**o**	**k**	**i**	0
3	_	**b**	**o**	**o**	**k**	**i**	**n**	0
4	**b**	**o**	**o**	**k**	**i**	**n**	**g**	0
5	**o**	**o**	**k**	**i**	**n**	**g**	_	1
6	**o**	**k**	**i**	**n**	**g**	_	_	0
7	**k**	**i**	**n**	**g**	_	_	_	0

Table 3.1: Instances with morphological segmentation classifications derived from the word **book|ing**. Denotation of the classification labels is as follows: 0 = no morpheme boundary; 1 = morpheme boundary.

3.1 Morphological segmentation

As described in Subsection 2.2.1, the subtask of morphological segmentation is deemed essential as the task that should be performed first in a word-pronunciation system, since knowledge about morpheme boundaries is relevant at other abstraction levels. Rather than implementing the knowledge generally assumed necessary (viz. knowledge of all morphemes occurring in English, of spelling changes at morpheme boundaries, and of the morphotactics of English, cf. Subsection 2.2.1), we present the subtask of morphological segmentation to the learning algorithms as a one-step classification task. This means that we convert a word into fixed-sized instances of which the focus letter is mapped to a class denoting a morpheme boundary decision. Applying the windowing method to the example word **book|ing** leads to the instances displayed in Table 3.1.

Morphological segmentation: Experiments

An instance base is constructed containing 675,745 instances, derived from the 77,565-word base (cf. Subsection 2.3). A problem with obtaining the morphological classifications is that CELEX does not provide complete information on the morphology of words. For about 10,000 words in the standard

77,565 data base, CELEX does not provide a morphological analysis, as these words fall in either of the following CELEX categories: (i) *irrelevant*, as in the onomatopoetic **meow**, or **gosh**; (ii) *obscure*, as in **tabby** which appears to stem from **tab**, but probably does not; (iii) *including a non-English root*, e.g., **imprimatur** and **patrimony**; and (iv) *undetermined*, mostly used for loan words such as **hinterland** and **virtuoso**. These unanalysed words may cause the learning algorithms ambiguity problems both during learning, when word pairs such as **cab|by** and **tabby** (unanalysed), or **farm|land** and **hinterland** (unanalysed) present counter-examples to each other, as well as during testing, when one of such a critical word pair is in the training set, and the other word is in the test set. We chose to include the unanalysed words in the data set to ensure comparability in data set size with the other subtasks investigated in this chapter. Lacking the morphological analyses for these words, derived instances are all associated with class 0.

The five learning algorithms BP, IB1, IB1-IG, IGTREE, and C4.5 are applied to the morphological-segmentation subtask. All algorithms are run using the default settings of Table 2.4. Figure 3.1 displays the generalisation errors obtained with these five algorithms in the bar graph displayed on the right. The left part of the figure displays a results summary, giving details on (i) the best-performing algorithm, (ii) the percentage of incorrectly-classified test instances by the best algorithm, and (iii) the percentage of incorrectly produced test words by the best algorithm. Below this results summary, the figure also lists the percentage of errors on test instances (i) when the overlap between test set and training set is taken as the only source of classification, and (ii) when the class bias (i.e., guessing the most-frequently occurring class in the data set) is taken as the source of classification. The generalisation accuracy of DC, displayed in terms of generalisation errors as the leftmost bar in the right part of Figure 3.1, is the combination of classification methods (i) and (ii) (cf. Subsection 2.4.2, p. 56). On top of each bar in the bar graph, an error bar displays the standard deviation (Hays, 1988) of the average value the bar represents. The value represented by each bar is also printed immediately above the respective bar.

IB1-IG performs significantly better than all other algorithms (the smallest difference is with IB1: $t(19) = 6.84, p < 0.001$). The difference in the accuracies of IB1 and IGTREE is not significant; all other differences are significant with $p < 0.001$. Thus, instance-based learning in IB1-IG appears most suited for learning morphological segmentation. While the generalisation accuracy of IB1 is matched by the decision-tree learning algorithm IGTREE, the use of the

RESULTS SUMMARY

best algorithm	IB1-IG
gen. error (%)	4.78
gen. error, words (%)	29.34

OVERLAP AND BIAS

overlap error (%)	21.31
class bias error (%)	24.56

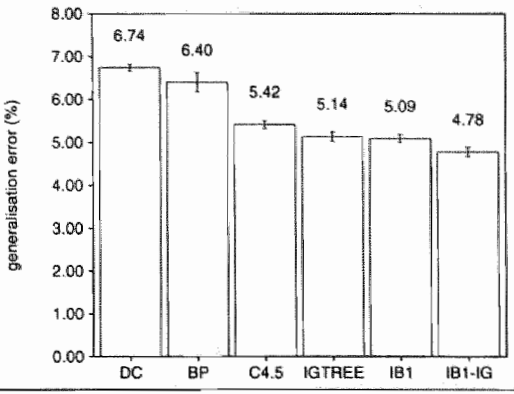


Figure 3.1: Results on the morphological-segmentation subtask. Results summary, bias, and overlap errors (left), and generalisation errors in terms of the percentage of incorrectly classified test instances of five algorithms (right).

information-gain-weighting function in IB1-IG is the decisive advantage in attaining the best accuracy.

The baseline accuracy score of DC (cf. Section 2.4, page 56) is quite close to the accuracies of the other algorithms. This can be explained by the fact that morphological boundaries are *regularly* located at word endings and beginnings, at which highly frequent suffixes such as **-ment** and **-ness**, and prefixes such as **pre-** and **un-** occur. Consequently, the instance base will contain many duplicates of letter windows capturing these frequent morphemes. As DC obtains its accuracy partly from the overlap between training and test sets, it benefits from these duplicates. On average, 82.8 % of the test instances overlap with training instances; of these overlapping instances, 96.2 % have a matching classification (the 3.8% defective cases are ambiguous instances, with identical feature values yet with different classifications). Thus, the relative success of DC reflects the fact that morphological segmentation benefits from storing instances in memory without abstraction by forgetting information (as in BP and the decision-tree algorithms). The generalisation accuracies of IB1 and IB1-IG of course corroborate this conclusion.

3.2 Graphemic parsing

The subtask of graphemic parsing can be paraphrased as 'identifying the letters or letter groups in a spelling word that map to a single phoneme in pronunciation'. The *raison d'être* of graphemic parsing as a discernible subtask within morpho-phonology is that before a phonemic transcription can be generated on the basis of a spelling word, the relations between graphemes (i.e., letters or letter groups) and phonemes have to be determined. This subtask is not trivial: it is a well-known fact of the English writing system that it has many multi-letter combinations, i.e., graphemes, realised in pronunciation as single phonemes (cf. Subsection 2.2.2).

CELEX does not contain information regarding the relation of letters and phonemes within words. Although CELEX contains the information that, for example, **booking** is pronounced /bukɪŋ/, it does not explicitly indicate the mapping between **oo** and /u/, or **ng** and /ŋ/. Consequently, a data set of graphemic parsing has to be compiled beforehand. One option to do this is to implement a rule system containing a limited number of context-sensitive rules describing the *graphotactics* (i.e., the rules and exceptions determining the possible orderings and co-occurrences of letters into graphemes) of English. However, we aim to minimise the inclusion of language-specific knowledge in both input and output of our tasks. This led us to develop a language-independent, data-oriented method (Daelemans and Van den Bosch, 1997). The algorithm attempts to make equal the length of a word's spelling string to the length of its transcription. Table 3.2 displays two examples of letter-phoneme alignments of the word **booking**. The top alignment represents a naive left-aligned parse that is clearly not optimal, as it maps letters to phonemes that are in no sensible way related (e.g., **o** and /k/, or **k** and /ɪ/). The bottom parse, actually produced by our automatic graphemic-parsing algorithm (Daelemans and Van den Bosch, 1997), presents an alignment that is at least intuitively correct.

In a first processing stage, the algorithm automatically captures letter-phoneme associations in an association matrix. Each spelling string is aligned to the left with its transcription of equal or shorter length. For each letter, the score of the phoneme that occurs at the same position in the transcription is incremented; furthermore, if a spelling string is longer than its transcription, phonemes which precede the letter position are counted as possibly associated with the target letter as well. In the case of the sample word **booking** (seven letters) with the unaligned phonemic transcription /bukɪŋ/ (five phonemes), for each letter, three phonemes receive a score increase: the phonemes that

bad parsing						
b	**o**	**o**	**k**	**i**	**n**	**g**
|	|	|	|	|	|	|
b	u	k	ɪ	ŋ	-	-
reasonable parsing						
b	**o**	**o**	**k**	**i**	**n**	**g**
|	|	|	|	|	|	|
b	u	-	k	ɪ	ŋ	-

Table 3.2: Examples of a bad letter-phoneme alignment (top) and a reasonable letter-phoneme alignment (bottom).

letter	no shift	shift + 1	shift + 2
b	b		
o	u	b	
o	k	u	b
k	ɪ	k	u
i	ŋ	ɪ	k
n		ŋ	ɪ
g			ŋ

Table 3.3: All possible associations between letters and phonemes of the sample word **booking** and its phonemic transcription /bukɪŋ/, shifted along the spelling.

are in the same position as the letters in focus, and the phonemes that may be in the same position when the phonemic transcription is shifted towards the right-alignment position (for the sample word, $7 - 5 = 2$ phonemes). Table 3.3 displays all possible letter-phoneme associations for the sample word **booking** (the empty cells in the table indicate word boundaries and do not count as phonemes)

Although a lot of noise is added to the association matrix by including associations that are less probable, the use of this association window ensures that the most probable associated phoneme is always captured in this window. The score of the phonemes is not increased equally for all positions: in the

present implementation of the algorithm (Daelemans and Van den Bosch, 1997) the focus phoneme receives a score increase of 8; the phonemes to the left receive a score increase of 4, 2, and 1 respectively; phonemes situated further in the string do not receive any score. When all words are processed this way, the scores in the association matrix are converted into probabilities.

The second part of the alignment algorithm generates for each pair of unaligned spelling and phoneme strings all possible (combinations of) insertions of null phonemes in the transcription. For each hypothesized string, a total association probability is computed by multiplying the scores of all individual letter-phoneme association scores between the letter string and the hypothesized phonemic string. The hypothesis with the highest total association probability is then taken as output of the algorithm.

A graphemic parsing of the letter string can be derived straightforwardly from the automatically-generated letter-phoneme alignments. In the example of **booking**, the reasonable alignment of Table 3.2 can simply be translated into a division of the word into the graphemes **b**, **oo**, **k**, **i**, and **ng**, when each letter mapping to '-' (i.e., the *null-phoneme*) is concatenated with its left neighbour letter to form a multi-letter grapheme). The null-phoneme is neither a member of the phonemic alphabet nor a (psycholinguistically) real element in human speech. It serves its purpose only in the application described here, and, e.g., in NETTALK (Sejnowski and Rosenberg, 1987)[1].

For the application of learning algorithms to the subtask of graphemic parsing, we construct a data set using the windowing method (cf. Subsection 2.3). Table 3.4 displays an example of the windowing conversion of the word **booking** to seven instances, using them graphemic parsing displayed in Table 3.2 (bottom). The spaces before and after words are represented by '_' characters. Each instance maps to either class 1, denoting that the focus letter is the first letter of a grapheme, or to class 0, denoting that the focus letter is *not* the first letter of a grapheme.

Graphemic parsing: Experiments

A data base is constructed of 675,745 instances derived from the 77,565-word base (cf. Subsection 2.3). BP, IB1, IB1-IG, IGTREE, and C4.5, using the default parameters listed in Table 2.4, are applied to this data base. Figure 3.2 displays

[1] Sejnowski and Rosenberg implicitly defend the use of phonemic nulls by assigning them the phonological feature *elide* (Sejnowski and Rosenberg, 1987). However, the corresponding phonological process of *elision* applies in considerably less cases (cf. Tranel, 1995) than the phonemic null occurs in the NETTALK data and in our data.

instance number	left context			focus letter	right context			class
1	_	_	_	**b**	**o**	**o**	**k**	1
2	_	_	**b**	**o**	**o**	**k**	**i**	1
3	_	**b**	**o**	**o**	**k**	**i**	**n**	0
4	**b**	**o**	**o**	**k**	**i**	**n**	**g**	1
5	**o**	**o**	**k**	**i**	**n**	**g**	_	1
6	**o**	**k**	**i**	**n**	**g**	_	_	1
7	**k**	**i**	**n**	**g**	_	_	_	0

Table 3.4: Example of applying the windowing encoding scheme to the word **booking** producing 7 instances that each map to a letter-phoneme alignment.

the generalisation error results of the algorithms, DC, and the overlap and bias errors. IB1-IG is found to produce significantly less errors on test instances than all other algorithms. The algorithm with the generalisation error closest to that of IB1-IG in terms of significance is BP ($t(19) = 13.84$, $p < 0.001$). IB1, IGTREE, and BP all perform roughly similarly: only IGTREE produces significantly less errors than IB1 ($t(19) = 2.81$, $p < 0.01$). The lazy-learning approach appears to result in optimal accuracy on the graphemic-parsing task only when it is combined with information-gain feature weighting in IB1-IG. Surprisingly, C4.5 performs significantly worse ($p < 0.001$) than all other algorithms except DC.

3.3 Grapheme-phoneme conversion

The conversion of graphemes to phonemes is a *transcription* task rather than a *segmentation* task such as morphological segmentation and graphemic parsing. As described in Subsection 2.2.2, the conversion of graphemes to phonemes constitutes a many-to-many mapping: a grapheme can map to many different phonemes, and a phoneme can be the transcription of many different graphemes.

The subtask of grapheme-phoneme conversion is defined as mapping a letter in a fixed-width window to either a phoneme or a phonemic null,

RESULTS SUMMARY

best algorithm	IB1-IG
gen. error (%)	1.05
gen. error, words (%)	7.60

OVERLAP AND BIAS

overlap error (%)	17.85
class bias error (%)	18.84

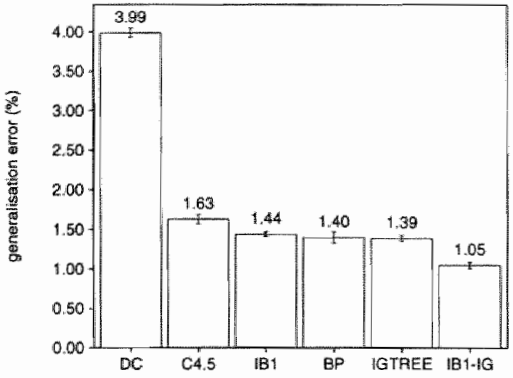


Figure 3.2: Results on the graphemic-parsing subtask. Results summary, bias, and overlap errors (left), and generalisation errors in terms of the percentage of incorrectly classified test instances of five algorithms (right).

the latter indicating that the letter is not pronounced. This is analogous to the **kn** task (cf. Section 2.1.1) in which the letter **k** either mapped to the phoneme /k/ or to the phoneme /-/ (the phonemic null). The inclusion of the phonemic null as possible classification, i.e., as the classification of letters that are part of a grapheme and are not pronounced, is an indirect way of encoding graphemic parsing in the grapheme-phoneme conversion task. By this subtask definition, the algorithms learn to map letters to phonemes, while also learning when to map letters not in graphemic-initial position to the phonemic null. Declaring the phonemic null to be a member of the phonemic alphabet is a solution to meet rule (iii) of the subtask definitions listed in the first section of this chapter (p. 60). Including the phonemic null, the grapheme-phoneme conversion subtask is to choose between 42 different phonemes (classes) when classifying instances.

To generate the instances for the grapheme-phoneme conversion task, we combine the phonemic transcription provided by CELEX with the output of the automatic graphemic-parsing algorithm used in the previous section (Daelemans and Van den Bosch, 1997), thereby designating this output to be flawless (as it is the only reasonable output available). Each instance that mapped to 1 in the graphemic-parsing subtask (Section 3.2) is mapped to its corresponding phoneme here. Each instance that mapped to 0 in the graphemic-parsing subtask, is mapped to a phonemic null '-'. The sample

instance number	left context			focus letter	right context			class
1	_	_	_	**b**	**o**	**o**	**k**	/b/
2	_	_	**b**	**o**	**o**	**k**	**i**	/u/
3	_	**b**	**o**	**o**	**k**	**i**	**n**	/-/
4	**b**	**o**	**o**	**k**	**i**	**n**	**g**	/k/
5	**o**	**o**	**k**	**i**	**n**	**g**	_	/ɪ/
6	**o**	**k**	**i**	**n**	**g**	_	_	/ŋ/
7	**k**	**i**	**n**	**g**	_	_	_	/-/

Table 3.5: Example of applying the windowing encoding scheme to the word **booking**, transcription /bukɪŋ/, introducing two null-phonemes placed beforehand by automatic graphemic parsing.

word **booking** is converted into grapheme-phoneme conversion instances in Table 3.5.

Grapheme-phoneme conversion: Experiments

An instance base is constructed containing 675,745 instances derived from the 77,565-word base. BP, IB1, IB1-IG, IGTREE, and C4.5 are applied to this instance base. Figure 3.3 displays the results obtained with the five algorithms, DC, and the overlap and bias errors. IB1-IG performs best on test instances. All differences between algorithms are significant with $p < 0.001$, except for the difference between IGTREE and C4.5 ($t(19) = 1.16, p \geq 0.05$). Measuring the accuracy in terms of correctly-processed test words rather than phonemes, IB1-IG can on the average transcribe flawlessly 79.9% of all test words, which is very close to high-quality standards demanded by industrial text-to-speech-synthesis developers, viz. 80%–90% flawless phonemic word transcriptions (Yvon, 1996). The high generalisation errors made by DC stem from the high number of classes in the phoneme task (viz. 62): guessing the most frequently occurring class, /-/, does not provide a good bias as it occurs only in 18.8% of the instances. The relatively high generalisation errors made by BP are harder to explain; the fixed number of hidden units (viz. 50) might be too low, causing difficulties in learning intermediary representations in the network (which has 294 input units and 62 output units).

RESULTS SUMMARY

best algorithm	IB1-IG
gen. error (%)	3.14
gen. error, words (%)	20.07

OVERLAP AND BIAS

overlap error (%)	19.87
class bias error (%)	81.16

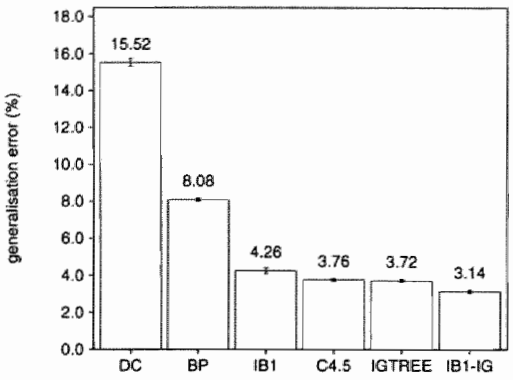


Figure 3.3: Results on the grapheme-phoneme-conversion subtask. Results summary, bias, and overlap errors (left), and generalisation errors in terms of the percentage of incorrectly transcribed phonemes of five algorithms (right).

instance number	left context			focus letter	right context			class
1	_	_	_	/b/	/u/	/k/	/ɪ/	0
2	_	_	/b/	/u/	/k/	/ɪ/	/ŋ/	0
3	_	/b/	/u/	/k/	/ɪ/	/ŋ/	_	1
4	/b/	/u/	/k/	/ɪ/	/ŋ/	_	_	0
5	/u/	/k/	/ɪ/	/ŋ/	_	_	_	0

Table 3.6: Instances derived from the syllabified phonemic string /bu-kɪŋ/.

3.4 Syllabification

The subtask of syllabification is to find the boundaries between syllables within strings of phonemes. We use the windowing method again to create fixed-sized instances. Instances map to class 1 when the focus phoneme is the first phoneme of a syllable, or to a class 0 when the focus phoneme is not the first phoneme of a syllable, or the first phoneme of the word (which is trivially at syllable-initial position when the word is pronounced in isolation). Table 3.6 lists the instances derived from the example phonemic string /bukɪŋ/, which is syllabified as /bu-kɪŋ/.

RESULTS SUMMARY

best algorithm	IB1-IG
gen. error (%)	0.40
gen. error, words (%)	2.22

OVERLAP AND BIAS

overlap error (%)	40.09
class bias error (%)	38.39

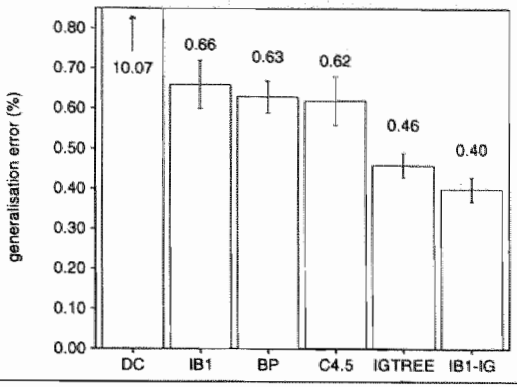


Figure 3.4: Results on the syllabification subtask. Results summary, bias, and overlap errors (left), and generalisation errors in terms of the percentage of incorrectly classified test instances of five algorithms (right).

Syllabification: Experiments

We construct an instance base of 548,417 instances[2] derived from 77,565 phonemic transcriptions. BP, IB1, IB1-IG, IGTREE, and C4.5 are applied to this instance base. The generalisation error results are displayed in Figure 3.4, as well as the generalisation accuracy of DC and the overlap and bias errors. All algorithms, except for the baseline DC, produce only a few classification errors on test instances. IB1-IG performs significantly better than all other algorithms (smallest difference is with IGTREE: $t(19) = 4.35$, $p < 0.001$). Non-significant differences exist between IB1, BP, and C4.5.

The accuracy with which the syllabification task is learned is in agreement with the overall regularity of the subtask according to phonological theories: syllabification is generally assumed to be governed by a few simple principles, if not only by one, i.e., the maximal onset principle (MOP) (cf. Section 2.2; Treiman and Zukowski, 1990). It appears that the regularity assumed by the MOP cannot be exploited by DC, which uses only overlap and bias to guess the classification of new instances; in contrast, the regularity is captured by each of the five learning algorithms.

[2]The number of instances in the syllabification instance base, 548,417, is smaller than that in the instance bases of morphological segmentation, graphemic parsing, and grapheme-phoneme conversion (viz., 675,745), because the phonemic transcriptions, excluding phonemic nulls, contain less phonemes than their respective words contain letters.

instance number	left context			focus letter	right context			class
1	_	_	_	/b/	/u/	/k/	/ɪ/	1
2	_	_	/b/	/u/	/k/	/ɪ/	/ŋ/	0
3	_	/b/	/u/	/k/	/ɪ/	/ŋ/	_	0
4	/b/	/u/	/k/	/ɪ/	/ŋ/	/_/	_	0
5	/u/	/k/	/ɪ/	/ŋ/	_	/_/	_	0

Table 3.7: Instances derived from the phonemic string with stress marker /'bukɪŋ/.

3.5 Stress assignment

In the CELEX data base of English, primary as well as secondary stress markers are provided as word stress information. These markers (' for primary stress, and " for secondary stress) are placed *before* the syllable receiving stress. For example, stress information for the word **booking** is stored as /'bu-kɪŋ/, which should be read as "primary stress falls on the first syllable /bu/". Using the windowing method, we generate phoneme string instances of which the focus phoneme maps to to class 1 if the focus phoneme is the first phoneme of the syllable receiving primary stress; to class 2 if the focus phoneme is the first phoneme of the syllable receiving secondary stress; and to class 0 otherwise. Every stress mark thus coincides with a syllable boundary (cf. Section 3.4). Table 3.7 lists the five instances derived from the phoneme string /bukɪŋ/.

Stress assignment: Experiments

We construct an instance base for English containing 548,417 instances, derived from 77,565 phonemic transcriptions. BP, IB1, IB1-IG, IGTREE, and C4.5 are applied to this instance base. The generalisation error results, the generalisation error of DC and the overlap and bias error are displayed in Figure 3.5.

The best generalisation accuracy is obtained with IB1-IG, which classifies 2.50% of the test instances incorrectly, on average. The difference between IB1-IG and IB1 (2.57% incorrect test instances) is small but significant ($t(19) = 2.80$, $p < 0.01$). All other differences between all algorithms are significant with $p < 0.001$. A remarkably bad accuracy is obtained with IGTREE. The information gain values of the features are relatively similar for this subtask (see Appendix D). Thus, the construction of trees by IGTREE is based on a

RESULTS SUMMARY

	IB1-IG
best algorithm	
gen. error (%)	2.50
gen. error, words (%)	15.14

OVERLAP AND BIAS

overlap error (%)	42.27
class bias error (%)	17.07

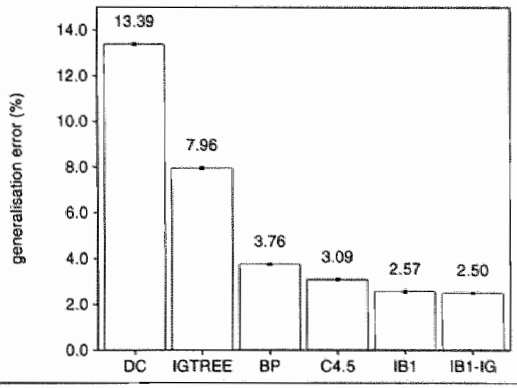


Figure 3.5: Results on the stress-assignment subtask. Results summary, bias, and overlap errors (left), and generalisation errors in terms of the percentage of incorrectly classified test instances of five algorithms (right).

feature ordering that is not very sound. Ignoring feature values, such as IGTREE does when building trees, constitutes a disadvantage in generalisation accuracy as compared to algorithms that do not ignore any feature values during classification, viz. IB1 and IB1-IG.

3.6 Chapter conclusion

Table 3.8 lists, for the five morpho-phonological subtasks investigated in this chapter, the generalisation errors of DC, BP, C4.5, IGTREE, IB1, and IB1-IG. The results summarise the superiority of IB1-IG on all subtasks. There are less clear differences between BP, C4.5, IGTREE, and IB1. IGTREE performs second best to IB1-IG on morphological segmentation, graphemic parsing, and syllabification. IB1 performs second best to IB1-IG on grapheme-phoneme conversion and stress assignment. Altogether, these results indicate that when inductive-learning algorithms are trained on subtasks of word pronunciation, they can generalise, with an amount of error that seems at least reasonable, to new instances of those subtasks. However, it is concededly arbitrary to consider generalisation errors below 5% (i.e., the highest error obtained with the best-performing algorithm, IB1-IG, on morphological segmentation) to be reasonable. We therefore concentrate on comparing the generalisation accuracies of algorithms.

task	algorithm					
	DC	BP	C4.5	IGTREE	IB1	IB1-IG
morphological segmentation	6.74	6.40	5.42	5.14	5.09	4.78
graphemic parsing	3.99	1.40	1.63	1.39	1.44	1.05
grapheme-phoneme conversion	15.52	8.02	3.76	3.71	4.69	3.10
syllabification	10.07	0.63	0.62	0.46	0.66	0.40
stress assignment	13.39	3.76	3.09	7.36	2.57	2.50

Table 3.8: Summary of the generalisation errors obtained with DC, BP, C4.5, IGTREE, IB1, and IB1-IG trained on five morpho-phonological subtasks.

Comparing the algorithms, the results strongly point to the superiority of lazy symbolic learning as displayed by IB1 and IB1-IG. They show the most suitable type of learning for obtaining the best generalisation accuracy on the five subtasks. The other three algorithms, BP, C4.5, and IGTREE, spend considerably more effort than IB1 and IB1-IG during learning in arriving at compressed knowledge representations, be it decision trees (IGTREE and C4.5) or real-valued intermediary representations between input and output in an MFN (BP). Abstraction of learning material by forgetting (data compression) can therefore claimed to be generally harmful for generalisation accuracy for our data.

A partial explanation for abstraction by forgetting being harmful to generalisation is the following. The full instance bases used for morphological segmentation, graphemic parsing, and grapheme-phoneme conversion, contain the same 675,745 letter-window instances when ignoring the classifications of the different subtasks. Although the words from which these instance bases are derived are all unique, only 224,677 unique instances occur (since words can be very alike in their spelling, e.g., in their endings). No less than 49.2% of these unique instances occur twice or more in the full instance base: in effect, 83.2% of the 675,745 instances in the full instance base have one or more identical instances. This percentage is, not surprisingly, equal to the average overlap between test sets and training sets as exploited directly by DC. Thus, storing all feature values of all training instances guarantees the exact matching of about 83.2% of the test instances (or the considerably lower 60.3% with the instance bases of syllabification and stress assignment) to the stored training instances. When an algorithm does not store all feature values

of all training instances, as is the case with induction of decision trees and backpropagation in MFNs, the guarantee to find overlapping instances is lost, hence generalisation accuracy may be lost.

The generalisation accuracy of an algorithm on all test instances also depends on the best-guess strategy employed to classify the 16.8% (on average) of the test instances which have no identical instance in the training set. A good strategy may undo the potential loss of generalisation caused by not remembering all feature-value information of training instances. For example, the strategy of storing default-class information on non-terminal nodes as employed by IGTREE appears sometimes to be more effective than the strategy based on counting class occurrences of best-matching instances employed by IB1 and IB1-IG: IGTREE generalises better than IB1 on graphemic parsing, grapheme-phoneme conversion, and syllabification.

In conclusion, keeping all feature values of all instances in memory guarantees the best generalisation accuracy only when instance-based (lazy) learning is employed *and* information gain is added as weighting function. The method of implementing word-pronunciation subtasks as classification tasks represented by instance bases containing fixed-size letter instances, touches on the apparent fact that there are considerable differences in the relevance of letter positions for the different tasks. The consistent advantage in generalisation accuracy of IB1-IG over IB1 suggests that it is profitable for generalisation accuracy when a learning algorithm takes this fact into account.

Chapter 4

Modularisation by sequencing subtasks

In this chapter we commence applying inductive-learning algorithms to the word-pronunciation task. Mainstream research on modelling this task (Allen *et al.*, 1987; Daelemans, 1987; Coker *et al.*, 1990; Van Heuven and Pols, 1993) has worked with the assumption of several necessary levels of abstraction between spelling and stressed-phonemic representations. We have highlighted five levels of abstraction in Section 2.2, and we have applied the five inductive-learning algorithms BP, IB1, IB1-IG, C4.5, and IGTREE to each of the five subtasks associated with these levels, i.e., morphological parsing (henceforth abbreviated as M), graphemic parsing (A), grapheme-phoneme conversion (G), syllabification (Y), and stress assignment (S). According to mainstream linguistic ideas, pronouncing words involves performing these subtasks in a particular *sequence*. To model word pronunciation in a system, one first has to model each subtask in a separate *module*. Second, one has to connect these modules in such a way that each module receives the correct type of input and produces the desired type of output that is either used as input to another module, or used as output of the whole system. We extend the idea of constructing a sequential-modular word-pronunciation system by using inductively-learned models as the modules. In Section 4.1 we describe our procedure for constructing such modular systems of which the modules are individually trained on their respective subtask.

In the literature, we find different suggestions for specific sequences of the five subtasks. For example, Allen *et al.* (1987) suggest that the sequence be M-A-G-Y-S. (By this acronym we mean that the five subtasks are performed in

the sequence starting with the leftmost subtask, morphological parsing (M); the hyphens indicate that the module immediately to the left of the hyphen provides the input to the module immediately to the right of the hyphen.) When a module A is placed before another module B in a modular sequence, the system designer assumes that the output of A is beneficial to learning to perform the subtask of module B. We provide some examples of the utility of using information produced by previous modules:

- The word **loophole** is composed of the morphemes **loop** and **hole**. Knowing that there is a morphological segmentation between **p** and **h** prevents the incorrect parsing of the grapheme **ph**. Knowing that **p** and **h** are both single-letter graphemes, it becomes highly unlikely that **p** or **h** are pronounced /f/. Having established that **p** and **h** are realised as /p/ and /h/, respectively, it is possible to determine that a syllable boundary lies between these phonemes (Treiman and Zukowski, 1990). Thus, output of M can serve as useful information to A, G, and indirectly to Y; output of G can serve as directly useful information to Y.

- The word **payee** has primary stress on the second syllable, while the word **pay** has primary stress on the first syllable. Having detected that **ee** is a morpheme, and knowing (or having induced) that words ending in **ee** almost always receive primary stress on the final syllable (because, linguistically speaking, **ee** is a stress-attracting affix), the correct stress can be predicted. Thus, output of M can serve as useful information to S.

- The word **through** is composed of three graphemes: **th**, **r**, and **ough**. This knowledge is essential for transcribing the spelling to the phonemic transcription: **th** maps to one phoneme (/θ/, not **t** and **h** separately; **r** maps to /r/; and **ough** maps to the single phoneme /u/ and not to two, three, or four phonemes. Thus, output from A can serve as useful information to G.

- The pronunciation of the word **behave**, /bəheɪv/, contains two syllables, /bə/ and /heɪv/; not three syllables such as /bə/, /heɪ/, and /vi /: the final **e** is not realised in pronunciation and thus cannot be the nucleus of a third syllable. Thus, the output of A and G can serve as useful input to Y. Furthermore, combining the facts that the phonemic transcription of **behave** contains two syllables and that **be** is a morpheme (a stress-neutral affix) which almost never receives stress, can lead to

the correct conclusion that the second syllable /heɪv/ receives primary stress. Thus, output of M, G and Y can serve as useful information to S.

In Section 4.2, two suggestions for sequential modular word-pronunciation systems are implemented as inductively-learned modules performing the subtasks in the designated sequential orderings, viz. M-A-G-Y-S and M-Y-S-A-G. Two algorithms, IGTREE and BP, are employed to train the individual modules. The reason for restricting our experiments to these algorithms is that only IGTREE and BP create relatively small models, which allows maintaining five-module systems in an average-sized computer memory[1]. IB1 and IB1-IG use very large instance bases; simultaneous storage of five instance bases in average-sized memory is not feasible. C4.5 was excluded for similar reasons; it generates overly large trees on the subtasks described in Chapter 3 (in particular on grapheme-phoneme conversion, due to an inefficiency in the tree generation procedure (Daelemans *et al.*, 1997a) which causes C4.5 to generate arcs for all possible feature values at all nodes of the tree).

The analysis in Subsection 4.2.3 of the two systems M-A-G-Y-S and M-Y-S-A-G focuses on their respective generalisation accuracies, and on the question which of the two systems represents the best ordering in terms of generalisation accuracy.

In Section 4.3 we investigate whether the two five-module systems investigated in Section 4.2 can both be reduced to two three-module systems. In the section, the hypothesis is formulated that graphemic parsing (A) and syllabification (Y) could be left implicit when learning grapheme-phoneme conversion (G) and stress assignment (S), respectively, and can thus be left out of the sequence of subtasks. This leads to the implementation of the two systems M-G-S and M-S-G, of which the modules are again trained with IGTREE as well as BP. In the analysis of results in Section 4.3 the generalisation accuracies of the M-G-S and M-S-G systems are compared. Furthermore, a comparison is made with the generalisation accuracies of the two three-module systems and the two five-module systems, providing indications on the influence of the number of modules on generalisation accuracy.

While Sections 4.2 and 4.3 report on generalisation accuracies of the four modular word-pronunciation systems, they do not address the issue of measuring the positive or negative effects of letting modules be dependent of output from other modules. In Section 4.4 this analysis is carried out by sys-

[1] "Average" computer memory, being a volatile concept, should be interpreted as 16 MB, which was a common quantity in 1996, but which is rapidly becoming less than average.

tematically investigating the *utility* of performing one subtask before another, comparing generalisation accuracies obtained with IGTREE.

Section 4.5 concludes the chapter by summarising the key findings of Sections 4.2, 4.3, and 4.4.

4.1 Procedures for constructing sequential modular systems

To construct a modular system one needs to know (a) which modules constitute the system; (b) in which order the modules are passed through; and (c) how information is passed between the modules. In our experiments, these three requirements are dealt with as follows.

ad (a) We have constructed modular systems of which the modules perform one of the five morpho-phonological subtasks described in Chapter 3: morphological segmentation (M), graphemic parsing (A), grapheme-phoneme conversion (G), syllabification (Y), and stress assignment (S).

ad (b) The order in which the modules are passed through is constrained to a sequential order. Only one module is working at a time, and every module is passed only once. The orderings of subtask in the systems investigated are the following: M-A-G-Y-S, M-Y-S-A-G, M-G-S, and M-S-G.

ad (c) The streams of information between modules are guided by two parameters governing each ordering A-B of two modules A and B, viz. the *pass filter* and the *propagation delay*:

1. The *pass filter* determines **what** is passed as input from module A to module B, and what is not. In the systems described here, it is invariably the case that all output of module A is presented as input to the subsequent module B in the ordering A-B. Moreover, it is allowed that the information used as input for a module A is passed along (i.e., copied) to serve as additional input to B. Thus, in the ordering A-B-C, the input of C always includes the output of B, and may also include the output of A passed along via B.
2. The *propagation delay* determines **when** the output of module A is presented as input to module B. The patience filter is needed to prevent modules receiving incomplete information as input. We employ a generic patience filter that allows for the information to

> pass from any module A to any module B only when all instances of a single word are processed by A; i.e., information is passed word-by-word along the modules.

In all our implementations of modular systems, learning is limited to the individual modules. This learning is performed analogous to the experiments described on the five morpho-phonological subtasks in Subsections 3.1, 3.2, 3.3, 3.4, and 3.5. The subtasks investigated in Chapter 3 were defined according to three rules (cf. page 60):

(i) instances are taken directly from the data extracted from CELEX;

(ii) the input consists of nothing more than letters or phonemes; and

(iii) the output consists only of classes belonging to the task itself.

Defining the subtasks within the four modular systems, rule (i) is maintained. Rule (ii) has to be dropped from this list for defining modular subtasks, as some modules in the modular systems explicitly receive both segmentational information and letters or phonemes. Rule (iii) is maintained.

There is, however, a problem with applying rule (i) in the context of learning a subtask in a modular system. The rule strictly implies that each module is trained on perfect input and perfect output, i.e., on input and output directly taken from the CELEX data. Thus, when applying rule (i) one adheres to the idea that when an appropriate task decomposition has been found, it is safe to train each module on its designated subtask using perfect data. However, there is no guarantee in a trained sequential-modular system that the actual data received by a module is of the same quality as the data used in the training of the module. None of the five morpho-phonological subtasks investigated in Chapter 3 are performed flawlessly by any algorithm. Thus, modules can be expected to generate some amount of error, most probably on unseen (test) data. Erroneous output from one module may well constitute unfamiliar or misleading input to the next module, possibly leading to yet another classification error by this next module. Errors may *cascade*: one error made by any module in the sequence may result in all other subsequent modules generating errors on the same data.

An adaptation of rule (i) alleviating the idea of maximal accuracy of the input during learning is the rule (i*) which states that the input should be produced by *a single source which is assumed to be reliable*. That source may be CELEX, but it may also be a previous module. When the errors generated by this module display regularities that can be picked up by the next module,

it is conceivable that the next module will be able to recognise and recover from these certain types of errors made by its predecessor(s). With rule (i*), modules can be said to *adapt* themselves to the output of the previous module.

To allow for comparison between both approaches to subtask definition, we have applied rules (i) and (i*), both in combination with rule (iii), to all experiments with modular systems. The variant of an experiment implementing rules (i) and (iii) is henceforth referred to as the *perfect* variant (after the assumed perfectness of the input and output data extracted from CELEX); the variant of an experiment implementing rules (i*) and (iii) is henceforth referred to as the *adaptive* variant (after the adaptation occurring between modules under this variant).

4.2 Modular systems with five modules

In this section we introduce two modular systems for word pronunciation. Both systems are composed of five modules to be passed through in sequential order. The architecture of both systems is inspired by existing word-pronunciation systems. The modular structure of the first system, M-A-G-Y-S, is inspired on the modular structure of the word-pronunciation subsystem for pronouncing unknown words of MITALK (Hunnicutt, 1976; Allen *et al.*, 1987). The modular structure of the second system, M-Y-S-A-G, is inspired by the word pronunciation subsystem of GRAFON-D (Daelemans, 1987; Daelemans, 1988).

The next two subsections describe the systems and their performances separately.

4.2.1 M-A-G-Y-S

The architecture of the M-A-G-Y-S system is inspired by SOUND1 (Hunnicutt, 1976; Hunnicutt, 1980), the word-pronunciation subsystem of the MITALK text-to-speech system (Allen *et al.*, 1987). When the MITALK system is faced with an unknown word[2], SOUND1 produces on the basis of that word a phonemic transcription with stress markers (Allen *et al.*, 1987). This word-pronunciation process is divided into the following five processing components:

1. *morphological segmentation*, attempting to detect affixes and inflections;

[2] When a word is 'unknown' to MITALK, it means that its morphological decomposition module and its part-of-speech tagger were not able to analyse the word (Allen *et al.*, 1987).

2. *graphemic parsing*, determining which letters or letter groups map to single phonemes;

3. *grapheme-phoneme conversion*, performing a context-sensitive rule-based mapping from letters or letter groups to phonemes;

4. *syllabification*, finding the syllable boundaries in the phonemic string, taking into account the affix and inflection boundaries found by the morphological component; and

5. *stress assignment*, synthesizing the output of the first three components and assigning primary, secondary, or no stress to syllable nuclei, guided by rules from metrical phonological theory (Halle and Keyser, 1971) as well as heuristics.

Whereas the description of SOUND1 (Hunnicutt, 1976; Hunnicutt, 1980; Allen *et al.*, 1987) is not in terms of modules, we interpret the decomposition of word pronunciation as expressed in SOUND1's division of word pronunciation into five sequential components as the modular structure of the M-A-G-Y-S system. This is where any further comparison with SOUND1 stops – we are not claiming to re-implement SOUND1, nor interpreting results obtained with M-A-G-Y-S to reflect the performance of SOUND1.

The M-A-G-Y-S architecture is visualised in Figure 4.1. This figure displays both the modules (sharp-edged boxes) and the representations taken as input and produced as output by the modules (curved-edged boxes).It can be seen that the representations include direct output from previous modules, as well as representations copied from earlier modules. For example, the stress-assignment module takes as input the syllable boundaries generated by the syllabification module, but also the phoneme string generated by the grapheme-phoneme-conversion module, and the morpheme boundaries generated by the morphological-segmentation module.

From the English CELEX data we use the standard word base of 77,565 unique pairs of words with their stressed phonemic transcriptions. From this data base, sub-databases are extracted for each of the subtasks of the M-A-G-Y-S system. M-A-G-Y-S is put to the test with IGTREE[3] and BP. The algorithms

[3] Since the input of some of the subtasks of the M-A-G-Y-S system contain feature values with considerably different value distributions (e.g., letters and segmentation markers), it is more appropriate to use gain ratio as the feature-weighting function in IGTREE and IB1 (Quinlan, 1993). All experiments henceforth reported with this type of input, with IGTREE and IB1, are performed with gain-ratio weighting.

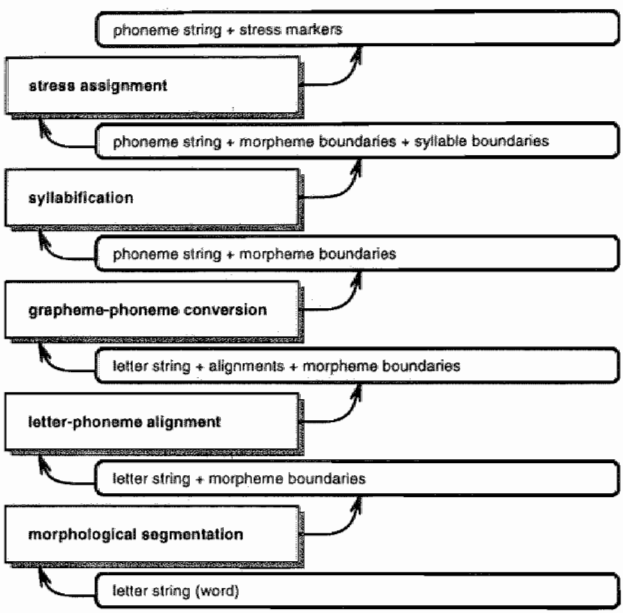


Figure 4.1: Visualisation of M-A-G-Y-S, a word pronunciation system containing five modules. Curved-edged boxes indicate input / output representations; sharp-edged boxes denote the five modules. The input-output mappings performed by the modules are depicted by arrows.

are executed using the standard settings of Table 2.4. For both algorithms, the perfect and the adaptive variants are tested. We thus perform four tests of the M-A-G-Y-S architecture: each test is conducted by running a 10-fold CV experiment on the full system. For the case of the perfect variant, this implies that during all ten experiments a fixed test set of 10% is withheld during the training of all modules. After training, the test set is processed sequentially through all modules of the system. The final output of the perfect variant of the M-A-G-Y-S system is the average output produced by the final module on the test sets over the ten experiments.

For the adaptive variant, the training and testing scheme is less trivial. A module can be trained only after the previous module has been trained; the source for its training and testing material is defined as "the output of the previous module", containing typical errors made by that module (cf. Section 4.1). To collect the typically erroneous output of a module, a data base is built containing the output of that module on all test words processed during the 10-fold CV experiment performed on the module. During each of the ten sub-experiments in the 10-fold CV experiment, the output produced on the 10% test words is concatenated to the data base. After the 10-fold CV experiment, the newly-formed data base contains as many items (words) as the data base used for training the module. The newly-formed data base is then partitioned into training and test sets again, for the 10-fold CV experiment on the next modular subtask. The average accuracy of the final module on its specific instance base (containing all cumulative performances of its four predecessors) over the 10-fold CV experiment with this instance base can be taken as the generalisation accuracy of the total M-A-G-Y-S system, as it expresses the accuracy of the total system as if the full database were processed as one large test set through all five modules in one pass.

Figure 4.2 displays the results obtained with IGTREE and BP on the perfect and adaptive variants of M-A-G-Y-S. A classification of an instance is regarded as incorrect if either or both of the phoneme and stress marker is incorrect.

The accuracies of BP and IGTREE on the perfect variant do not differ significantly ($t(19) = 0.66$, $p \geq 0.05$). However, significant differences are found between the perfect and adaptive variants of BP ($t(19) = 4.55$, $p < 0.001$), as well as between the perfect and adaptive variants of IGTREE ($t(19) = 38.75$, $p < 0.001$). Moreover, the accuracy of IGTREE under the adaptive variant is significantly better than BP under the adaptive variant ($t(19) = 27.94$, $p < 0.001$). Apparently, IGTREE can take good advantage of the adaptive strategy; the outputs of modules in the IGTREE-trained M-A-G-Y-S system do appear to contain

RESULTS SUMMARY

best algorithm	IGTREE (adaptive)
gen. error (%)	10.59
gen. error, words (%)	54.44

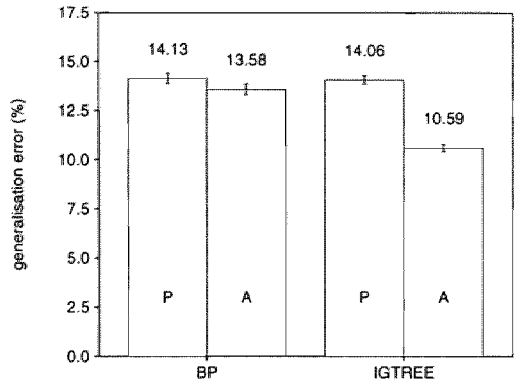


Figure 4.2: Results on the M-A-G-Y-S task. Results summary (left), and generalisation errors in terms of the percentage of incorrectly classified test instances by IGTREE and BP on the perfect (P) and adaptive (A) variants (right).

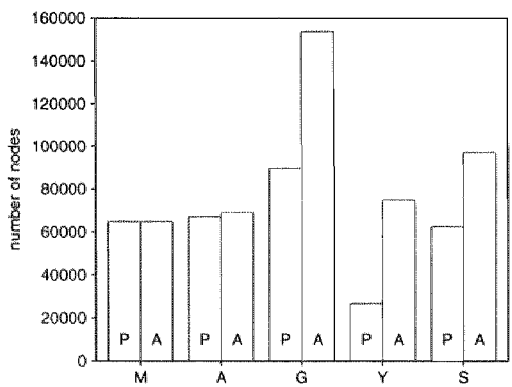


Figure 4.3: Average numbers of nodes in the decision trees generated by IGTREE for each of the five modules of the M-A-G-Y-S system, trained on the perfect (P) and adaptive (A) variants of the M-A-G-Y-S task.

typicalities that can be exploited in learning the tasks of consecutive modules. BP appears to lack the ability to recognise typical errors with the same success as IGTREE. One cause of BPs

Exploiting typicalities in errors of previous modules by IGTREE may lead to larger trees, as these trees have to represent both the regular classifications and the typical erroneous ones. This is confirmed by Figure 4.3 which displays the average number of nodes of the decision trees created by IGTREE for the perfect and adaptive variants. IGTREE builds larger trees when trained under the adaptive variant than when trained under the perfect variant. With the larger trees, better generalisation accuracy is obtained.

4.2.2 M-Y-S-A-G

The architecture of M-Y-S-A-G, displayed in Figure 4.4, is inspired by the architecture of GRAFON-D (Daelemans, 1987; Daelemans, 1988), a text-to-speech system for the Dutch language. In the word-pronunciation subsystem of GRAFON-D, a word is analysed by the following four components:

1. *morphological segmentation*, searching for an optimal segmentation of a word into morphemes using a two-step *generate-and-test* process;

2. *syllabification*, syllabifying the morphologically analysed word;

3. *stress-assignment*, placing stress markers on the syllabified, morphologically analysed word; and

4. *transliteration mapping and phonological rules*, synthesizing the information on syllable and morpheme boundaries and stress placement into a phonemic transcription.

Note that the description of the last component does not separate letter-phoneme conversion into graphemic parsing and grapheme-phoneme conversion. This is because in the GRAFON-D transliteration component graphemes are not taken as the basic input unit for grapheme-phoneme conversion, but rather the (groups of) letters that make up the onsets, nuclei, and codas of syllables. Transliteration to phonemes is performed on these letters or letter groups. For our M-Y-S-A-G system we explicitly abstract away from this transliteration strategy, and model the transliteration employing the modules used in the M-A-G-Y-S system, viz. graphemic parsing and grapheme-phoneme conversion. This choice allows us to compare the M-A-G-Y-S and M-Y-S-A-G systems by letting both systems be composed of the same modules. As with M-A-G-Y-S, we are not claiming to re-implement GRAFON-D,

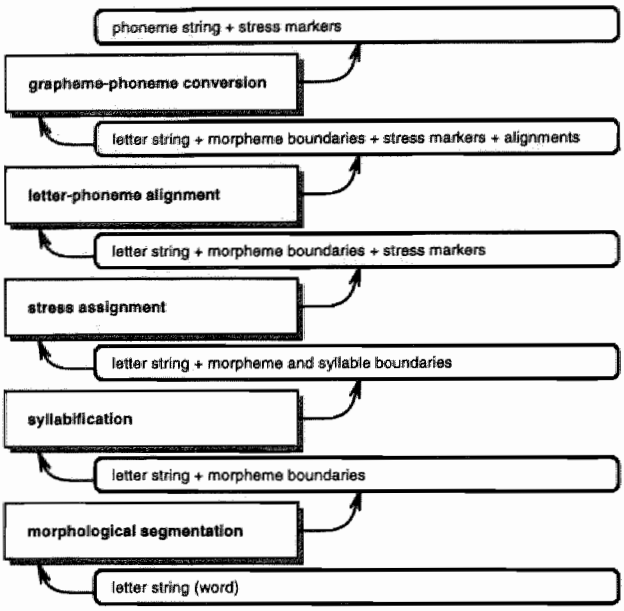


Figure 4.4: Visualisation of M-Y-S-A-G, a word-pronunciation system containing five modules. Round-edged boxes indicate input–output representations; sharp-edged boxes denote the modules. The input–output mappings performed by the modules are depicted by arrows.

nor interpreting the results obtained with M-Y-S-A-G as performance results of GRAFON-D. The assumption underlying GRAFON-D that we do adopt explicitly in M-Y-S-A-G is that syllabification and stress assignment should be performed before grapheme-phoneme conversion, rather than vice versa.

The experiments on the M-Y-S-A-G architecture are performed analogous to the experiments with M-A-G-Y-S reported in Subsection 4.2.1. Figure 4.5 displays the generalisation errors of IGTREE and BP on the perfect (P) and adaptive (A) variants of the M-Y-S-A-G task. The difference in accuracy between the two variants (P and A) is significant, notably for IGTREE ($t(19) = 47.25$, $p < 0.001$) but also for BP ($t(19) = 6.13$, $p < 0.001$). Again, as with M-A-G-Y-S, a better generalisation accuracy is obtained with the adaptive variant, and is the best accuracy obtained with IGTREE under this variant (significantly better than BP, $t(19) = 17.67$, $p < 0.001$).

RESULTS SUMMARY

best algorithm	IGTREE (adaptive)
gen. error (%)	11.29
gen. error, words (%)	60.14

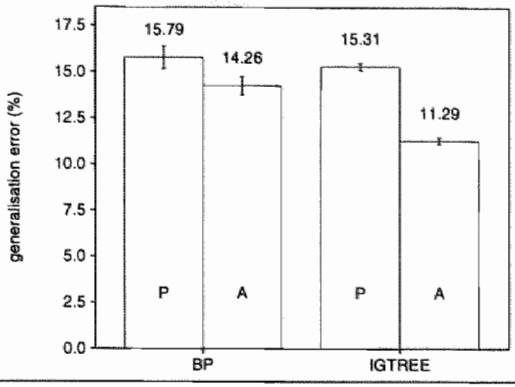


Figure 4.5: Results on the M-Y-S-A-G task. Results summary (left) and generalisation errors in terms of the percentage of incorrectly classified test instances by IGTREE and BP on the perfect (P) and adaptive (A) variants (right).

Figure 4.6 displays the average number of nodes of the decision trees created by IGTREE for the five modules of the M-A-G-Y-S system under the perfect and adaptive variants. Analogous to Figure 4.3, Figure 4.6 indicates that IGTREE builds larger trees for the modular subtasks under the adaptive variant as compared to the perfect variant. The figure shows that especially the trees built for the grapheme-phoneme-conversion module are much larger for the adaptive variant than for the perfect variant. Figure 4.3 displayed a similar phenomenon for the grapheme-phoneme-conversion module in the M-A-G-Y-S system; with the M-Y-S-A-G system, trees built for this subtask are even larger (viz. 210,509 nodes on average) than with the M-A-G-Y-S system (viz. 153,678 nodes on average). These results suggest that under the adaptive variant the trees constructed by IGTREE become increasingly bigger than their counterpart trees under the perfect variant as the module has more predecessor modules. More predecessing modules may generate more errors; unresolved errors are propagated further, and the total amount of unresolved propagated errors appears to increase roughly with each module in the sequence. The G-module in M-A-G-Y-S receives unresolved propagated errors from two modules, and needs 64,183 more nodes than under the perfect variant. The G-module in M-Y-S-A-G receives unresolved propagated errors from four modules, and needs 103,161 more nodes. The surplus of effort put into building trees by IGTREE under the adapted variant is nevertheless profitable

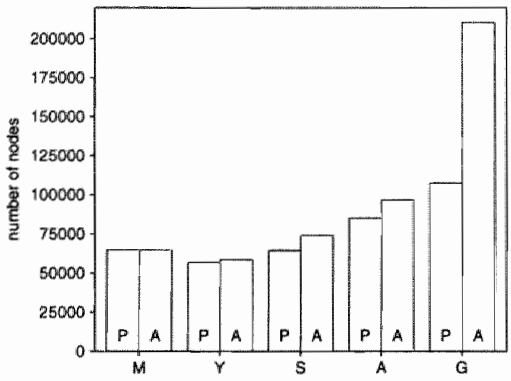


Figure 4.6: Average numbers of nodes in the decision trees generated by IGTREE for each of the five modules of the M-Y-S-A-G system, trained on the perfect (P) and adaptive (A) variants of the M-Y-S-A-G task.

for generalisation accuracy: as with M-A-G-Y-S, generalisation accuracy of the M-Y-S-A-G system constructed under the adaptive variant is significantly better than generalisation accuracy under the perfect variant (cf. Figure 4.5).

4.2.3 Comparing M-A-G-Y-S and M-Y-S-A-G

Figure 4.7 summarises the best results obtained with both systems. The figure shows a small but significant difference ($t(19) = 8.46$, $p < 0.001$) between the accuracies on the M-A-G-Y-S and M-Y-S-A-G tasks, with the former system displaying the best generalisation accuracy. Performing graphemic parsing and grapheme-phoneme conversion *before* syllabification and stress assignment thus leads to better word-pronunciation accuracy than performing the pairs of tasks in reverse order.

The actual accuracies of both systems are not impressive: the M-A-G-Y-S system classifies 10.6% of all test patterns incorrectly, the M-Y-S-A-G system 11.3%. In terms of the percentage of flawlessly processed test words (i.e., test words of which the phonemic transcription with stress markers does not contain any flaws in either the phonemes or the stress markers) the M-A-G-Y-S system reaches a disappointing score of 45.6%; M-Y-S-A-G produces only 39.9% of the test words flawlessly.

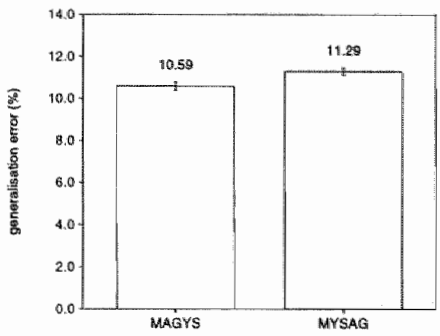


Figure 4.7: Generalisation errors of IGTREE, under the adaptive variant, applied to the two modular systems M-A-G-Y-S and M-Y-S-A-G.

4.3 Modular systems with three modules

In Section 3.3 we have shown that it is possible to integrate graphemic parsing with grapheme-phoneme conversion, by introducing phonemic nulls as the mapping for letters which are not pronounced. This subtask definition of grapheme-phoneme conversion is also used in the NETTALK model (Sejnowski and Rosenberg, 1987). It is therefore plausible to integrate the graphemic-parsing module (A) and the grapheme-phoneme module (G) into a single grapheme-phoneme conversion module G.

A similar argument can be made for integrating the syllabification and stress assignment modules into a single stress-assignment module. We remarked in Section 3.5 that stress markers are placed solely on the positions which are also marked as syllable boundaries (i.e., on syllable-initial phonemes). Removing the syllabification subtask makes finding those syllable boundaries which are relevant for stress assignment an integrated part of stress assignment. Syllabification (Y) and stress assignment (S) could be integrated in a single stress-assignment module S.

When both pairs of modules are reduced to single modules, two three-module systems are obtained. The first system, M-G-S, is derived from M-A-G-Y-S. The second system, M-S-G, stems from M-Y-S-A-G. Experiments with both systems are described in the subsequent Subsections 4.3.1 (M-G-S) and 4.3.2 (M-S-G). A comparison between their accuracies and their five-module counterparts is made in Subsection 4.3.3.

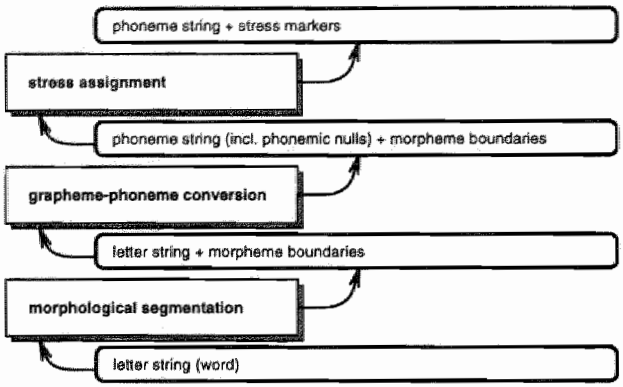


Figure 4.8: Visualisation of M-G-S, a word-pronunciation system containing three modules. Round-edged boxes indicate input / output representations; sharp-edged boxes denote the modules. The input–output mappings performed by the modules are depicted by arrows.

4.3.1 M-G-S

Figure 4.8 displays the architecture of the M-G-S system. When compared to the M-A-G-Y-S system (Figure 4.1), we see that neither graphemic parsings nor syllable boundaries are passed as information between modules, since the subtasks associated with these types of morpho-phonological information are integrated in the grapheme-phoneme conversion module and the stress-assignment module, respectively. Experiments on this system are performed analogous to the experiments with the M-A-G-Y-S and M-Y-S-A-G systems.

The generalisation errors of IGTREE and BP under both the perfect and adaptive variants are displayed in Figure 4.9. The figure shows a better accuracy of IGTREE than of BP. Regardless of the training variant, BP clearly suffers from bad generalisation accuracy of the system. In fact, the accuracies obtained with BP on M-G-S are hardly different from those obtained with BP on M-A-G-Y-S (cf. Figure 4.2). For IGTREE, however, a considerable improvement over its accuracy on M-A-G-Y-S is observed. On the adaptive variant the difference is significant ($t(19) = 37.50$, $p < 0.001$). Furthermore, the adaptive variant of IGTREE on M-G-S yields a significantly better accuracy than the perfect variant ($t(19) = 29.49$, $p < 0.001$).

As was shown in Figure 4.3, IGTREE created larger trees in the M-A-G-Y-S system for the adaptive variant as compared to the trees created for the perfect

RESULTS SUMMARY

best algorithm	IGTREE (adaptive)
gen. error (%)	7.86
gen. error, words (%)	55.11

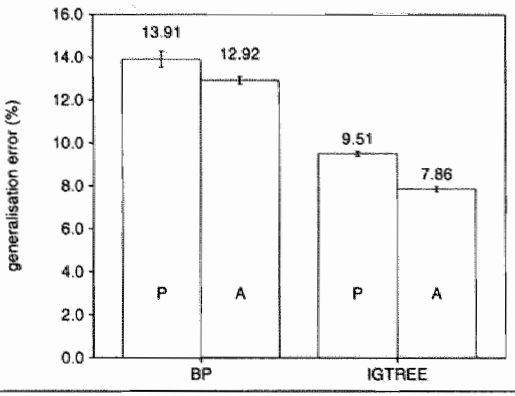


Figure 4.9: Results on the M-G-S task. Results summary (left), and generalisation errors in terms of the percentage of incorrectly classified test instances by IGTREE and BP on the perfect (P) and adaptive (A) variants (right).

variant. Figure 4.10 displays the total numbers of nodes of the trees generated for each of the three modules of the M-G-S system. The differences between the magnitudes of the trees are small, unlike the differences in tree sizes in the M-A-G-Y-S system. Only the final module (viz. stress assignment) of the M-G-S system trained under the adaptive variant needs a significantly larger tree than in the case of the perfect variant ($t(19) = 52.08, p < 0.001$). These results show that the M-G-S task constitutes a better modularisation than the M-A-G-Y-S task.

4.3.2 M-S-G

The system architecture for M-S-G is shown in Figure 4.11. M-S-G stems from M-Y-S-A-G: it replaces the syllabification and stress-assignment modules by one stress assignment module S, and replaces the graphemic-parsing module and grapheme-phoneme-conversion module by a single grapheme-phoneme module G.

The overall generalisation errors of IGTREE and BP are displayed in Figure 4.12. The figure displays analogous differences between BP and IGTREE under the two variants as Figure 4.9 does for the M-G-S system. Yet again, the results obtained under the adaptive variant are significantly better than those obtained under the perfect variant, for BP ($t(19) = 8.33$, $p < 0.001$) and IGTREE

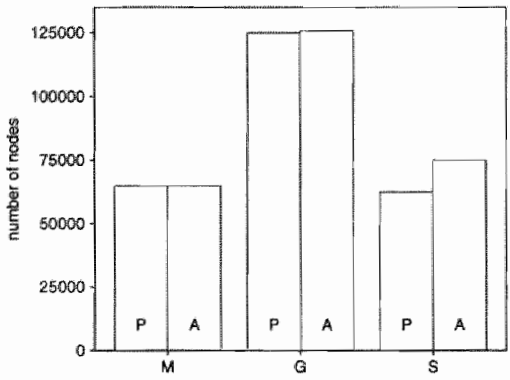


Figure 4.10: Average numbers of nodes in the decision trees generated by IGTREE for each of the three modules of the M-G-S system, trained on the perfect (P) and adaptive (A) variants of the M-G-S task.

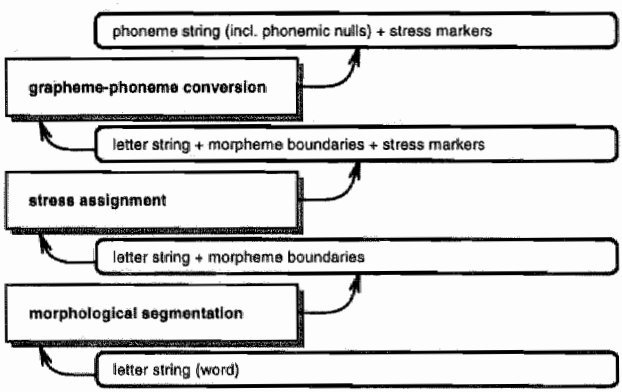


Figure 4.11: Visualisation of M-S-G, a word-pronunciation system containing three modules. Round-edged boxes indicate input–output representations; sharp-edged boxes denote the modules. The input–output mappings performed by the modules are depicted by arrows.

RESULTS SUMMARY

best algorithm	IGTREE (adaptive)
gen. error (%)	8.41
gen. error, words (%)	57.70

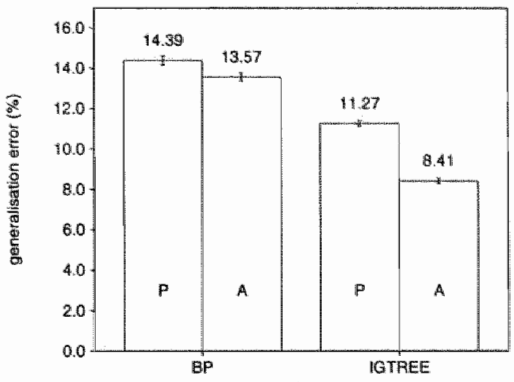


Figure 4.12: Results on the M-S-G task. Results summary (left) and generalisation errors in terms of the percentage of incorrectly classified test instances by IGTREE and BP on the perfect (P) and adaptive (A) variants (right).

($t(19) = 45.68$, $p < 0.001$). The accuracy of IGTREE is significantly better than that of BP under the adaptive variant ($t(19) = 64.65$, $p < 0.001$). Moreover, IGTREE performs better on the M-S-G task than on M-Y-S-A-G under the adaptive variant ($t(19) = 39.94$, $p < 0.001$), again indicating that decomposition of the word-pronunciation task in the three-module systems is better learnable for inductive-learning algorithms.

Figure 4.13 displays the amount of nodes in the trees constructed by IGTREE under the perfect and adaptive variants. In contrast with the previous figures displaying tree sizes under the perfect and adaptive variants (Figures 4.3, 4.6, and 4.10), the figure shows one occurrence of a slight decrease in average tree size on a single module trained under the adaptive variant, as compared to the perfect variant. The stress-assignment (S) module trained under the adaptive variant leads to trees with 78,322 nodes on average, while training under the perfect variant leads to trees with 80,402 nodes on average. It appears that there are typical errors in the output of the M-module that can be recognised and exploited favourably by the S-module. In sum, the M-S-G modularisation is better than the M-Y-S-A-G both in terms of the total number of nodes needed (cf. Figure 4.6) and in the amount of propagated errors in the system.

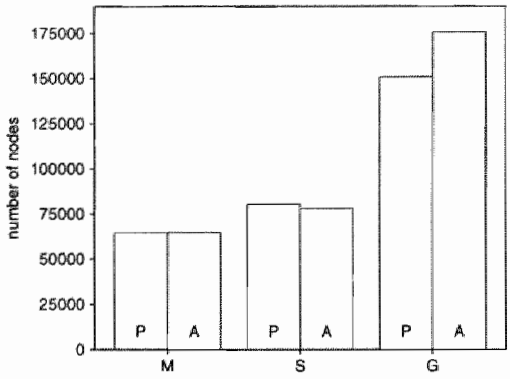


Figure 4.13: Average numbers of nodes in the decision trees generated by IGTREE for each of the three modules of the M-S-G system, trained on the perfect (P) and adaptive (A) variants of the M-S-G task.

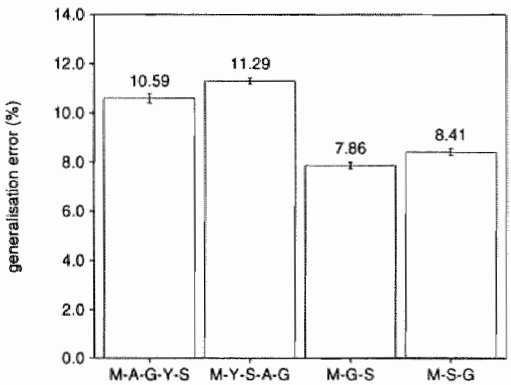


Figure 4.14: Generalisation errors of IGTREE, under the adaptive variant, applied to the four modular systems M-A-G-Y-S, M-Y-S-A-G, M-G-S, and M-S-G.

4.3.3 Comparing three-module and five-module systems

Figure 4.14 displays the results obtained from applying IGTREE under the adaptive variant to each of the four systems. These lowest error results reflect the best accuracies obtained with IGTREE for each of the systems.

The difference in generalisation accuracy between M-A-G-Y-S and M-G-S is distinctly large, and turns out to be significant ($t(19) = 37.50$, $p < 0.001$), as is

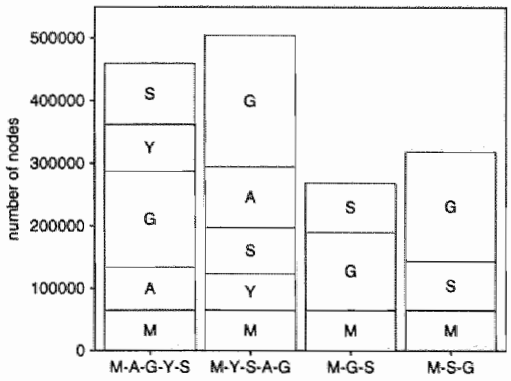


Figure 4.15: Average numbers of nodes in the decision trees generated by IGTREE for each of the four modular systems (trained under the adaptive variant).

the case with the difference between M-Y-S-A-G and M-S-G ($t(19) = 47.67, p < 0.001$). Thus, the three-module systems perform better than their five-module counterparts. Using three modules as opposed to five considerably reduces the amount of errors on test material.

Another advantageous difference between three-module systems and five-module systems is the total amount of memory needed for storing the trees. Figure 4.15 displays the summed number of nodes for each of the four IGTREE-trained systems under the adaptive variant. Each bar is divided into compartments indicating the amount of nodes in the trees generated for each of the modular subtasks.

In Subsections 4.3.1 and 4.3.2 we noted that IGTREE constructed larger M-A-G-Y-S systems than M-G-S systems, and larger M-Y-S-A-G systems than M-S-G systems; for reasons of comparison this result is again included in Figure 4.15. The figure also provides indications of the difference between M-A-G-Y-S and M-Y-S-A-G, and between M-G-S and M-S-G. This difference relates directly to the different ordering of grapheme-phoneme conversion and stress assignment, but it is dependent of the algorithm by which they are produced (IGTREE, under the adaptive variant). The compartments displayed in the bars of Figure 4.15 show that the difference is mostly due to grapheme-phoneme conversion taking up a disproportionally large part of the total amount of nodes in both the M-Y-S-A-G and the M-S-G system, i.e., when grapheme-phoneme conversion is preceded by stress assignment.

Figures 4.14 and 4.15 show that the model with the best accuracy, M-G-S, is also the model taking up the smallest number of nodes. Differences in accuracy, displayed in Figure 4.14, roughly correlate with differences in summed tree sizes: the smaller the total number of nodes, the better the generalisation accuracy. The statement that abstraction by compression is generally harmful for generalisation accuracy does not apply here: however, the latter statement refers to different algorithms applied to the same (sub)task. Here, results show that different definitions of the same task may allow one learning algorithm employing compression to obtain better generalisation accuracy with smaller models. The improvements in generalisation accuracy and size of induced models can be taken as indications for the appropriateness of the particular task definition: M-G-S is the most appropriate task definition (i.e., the definition leading to the best generalisation accuracy) of the four alternatives.

4.4 The utility of sequential modularisation

When the output of a module is presented as input to another module within a modular system, it is assumed that this input is useful for the classification subtask performed by the module. The sequences of the modules and their connectivity in a modular system reflect the assumptions of the system designer on the utilities of the various modules in the system. Below we analyse which of these assumptions are justified by examining the experimental performance results of the inductively-learned subtask modules. The results obtained with each module contain a large amount of rather opaque information, since the results are affected by four factors: (i) which other modules precede the module (hence, of which modules input is received), (ii) the order of these preceding modules, (iii) the way in which the system is learned, viz. under the perfect or adaptive variant, and (iv) which learning algorithm is used. Given these factors, it is hard, if not impossible, to determine exactly what the utility of a single modular input is to a certain module, since this effect is a merge of different effects. Only the net effect (i.e., the module's generalisation accuracy) is known. We rule out the factors (iii) and (iv) mentioned, by only investigating modules in systems trained with IGTREE under the adaptive variant, as this combination produces the best generalisation accuracies.

To measure the effect of including the output of module A in the input of module B, we compute a *utility effect* expressing the difference between the

generalisation accuracy of module *B* *without* the output of module *A* in the input, and the generalisation accuracy of module *B* *with* the output of module *A* in the input. A positive utility effect implies that the inclusion of the output of module *A* as input to module *B* has lead to a lower generalisation accuracy on the subtask performed by *B*: including the output of module *A* in *B*'s input is profitable. A negative utility effect indicates that including *A*'s output in *B*'s input is not profitable for generalisation accuracy.

The utility of the morphological-segmentation (M) subtask on the A, G, Y, and S tasks can be computed by subtracting the generalisation error obtained on the four corresponding isolated subtasks (described in Subsections 3.2 to 3.5) from the generalisation error of the second modules in the M-A-G-Y-S, M-G-S, M-Y-S-A-G, and M-S-G systems, respectively. For example, for graphemic parsing, the generalisation error of IGTREE on the isolated graphemic-parsing subtask is 2.39%, while the error on the A-module in the M-A-G-Y-S system (trained with IGTREE under the adaptive variant) is 2.50%. The utility effect is therefore 2.39 − 2.50 = -0.11. Since the utility effect is negative, it can be concluded that including the output of the morphological-segmentation module as input in the graphemic-parsing subtask is not profitable for the generalisation accuracy on the latter task.

Computing the utility effect of a module different from morphological segmentation is less straightforward. Suppose that we want to compute the utility effect of including the output of module *B* as input in module *C*, when neither *B* nor *C* perform morphological segmentation. In the case of an M-*B*-*C* system, the input to *C* includes the output of *B*, but also the output of M. Moreover, the output of M is also included in the input of *B*. When the accuracy of *C* in M-*B*-*C* is subtracted from the accuracy of *C* performed in isolation, it is not possible to interpret the resulting difference as a utility effect of *B* only. The chosen alternative to compute the utility of *B* as input to *C* is to compute the difference between the generalisation error of *C* in M-*B*-*C* and that of *C* in M-*C*-.... The resulting utility effect can only be interpreted when taking into account that both modules *B* and *C* receive input from M.

Table 4.1 lists all computed utility effects. Cells in the Table represent the utility effect of including the output of the module in the corresponding row, as input in the module in the corresponding column. An empty cell occurs when (i) the module in the row is identical to the module in the column, (ii) the module in the row is never the immediate predecessor of the module in the column (e.g., A never occurs immediately before Y), or (iii) more modules precede the module in the row and the module in the column than just M (e.g.,

module	subsequent module A	G	Y	S
M	−0.11	−0.27	−1.01	+3.49
A		−4.60		
G				+0.60
Y				−0.96
S		−0.48		

Table 4.1: Overview of utility effects of including the output of modules (in rows) as input to subsequent modules (in columns) in sequential modular systems, trained with IGTREE under the adaptive variant. A positive number in a cell indicates a better accuracy of the module in the column when the output of the module in the row is included in the input, as opposed to when it is excluded from the input.

S occurs immediately before A, but only in M-Y-S-A-G; the Y module preceded the S module, and there is no M-Y-A-G system to compare results with).

The utility of morphological segmentation

As a first analysis, we investigate the effect of morphological boundaries as extra input to the modules placed immediately after the morphological-segmentation module of all four modular systems, i.e., graphemic parsing in M-A-G-Y-S, grapheme-phoneme conversion in M-G-S, syllabification in M-Y-S-A-G, and stress assignment in M-S-G. The input these modules receive contains letters augmented with the output of the morphological segmentation module, which is the same for all systems, and which does not contain any propagated errors from any other modules.

The results in the first row of Table 4.1 show that morphological boundaries, yet containing errors, constitute useful input only to the stress-assignment module (this difference is highly significant, $t(19) = 127.19$, $p < 0.001$). Interestingly, no positive effect is measured on graphemic-parsing in the M-A-G-Y-S system, nor on grapheme-phoneme-conversion in the M-G-S system, nor on the syllabification task in the M-Y-S-A-G system. A part of the negative effect of including morpheme boundaries can be attributed to the errors of the morphological-segmentation module. Nevertheless, the negative effect on syllabification in M-Y-S-A-G suggests that morphology does

not play any role in syllabification, which is in accordance with mainstream phonological theories (Selkirk, 1984; Kenstowicz, 1993).

The utility of graphemic parsing

Graphemic parsing as a separate subtask occurs in the M-A-G-Y-S and M-Y-S-A-G systems. The assumption underlying both systems is that aligning phonemes with letters is a necessary subtask that needs to be performed before the conversion from graphemes to phonemes is performed. In the M-S-G and M-G-S systems it is assumed that it is better to incorporate graphemic parsing directly in the output of the grapheme-phoneme-conversion task. These two assumptions can be tested by comparing the generalisation accuracy on the grapheme-phoneme conversion task between M-A-G-Y-S and M-G-S, as argued earlier. The results displayed in the second row of Table 4.1 show that the placement of graphemic parsing as a separate module before grapheme-phoneme conversion leads to drastically worse accuracy on the latter module. The graphemic-parsing module apparently produces errors that have a profound negative effect on the accuracy of the grapheme-phoneme-conversion module. The results suggest that it is better to intergrate graphemic parsing with grapheme-phoneme conversion than be performed as a separate subtask before grapheme-phoneme conversion when learning these tasks with IGTREE.

The utility of grapheme-phoneme conversion

Grapheme-phoneme conversion produces phonemes as output. While these phonemes are essential for constituting half the word-pronunciation output, they are also assumed to be useful as input to stress assignment in the M-A-G-Y-S and M-G-S systems. An analysis of the utility of grapheme-phoneme conversion should therefore focus on the comparison of the generalisation accuracies on stress assignment in the two three-module systems. The accuracy difference, listed in the third row of Table 4.1, indicate that inclusion of the output of the grapheme-phoneme-conversion task in the input of the stress assignment task (i.e., phonemes and morpheme boundaries rather than letters and morpheme boundaries) improves the generalisation accuracy of IGTREE on the latter task by a significant margin ($t(19) = 16.27$, $p < 0.001$). This means that IGTREE applied to stress assignment with morpheme boundaries and phonemes as input (including erroneous phonemes), performs better than IGTREE applied to stress assignment on the basis of morpheme bound-

aries and letters. As stated earlier, this utility effect is dependent of the morphological-segmentation module.

The utility of syllabification

The effect of including the syllabification subtask as a module before stress assignment can be tested in a way analogous to the utility of graphemic parsing, by comparing the accuracy on stress assignment in the pair M-Y-S-A-G and M-S-G. In M-Y-S-A-G, syllabification is assumed to provide useful information for the stress assignment module. It was argued earlier in Section 3.5 that, as stress markers are placed on syllable boundaries, it is useful to know the place of the syllable boundaries before performing stress assignment. The accuracy differences listed in Table 4.1 indicate that this is not the case, at least not with the additional input from the morphological-segmentation module. The syllabification module, which was found to suffer from input from the morphological-segmentation module (viz. -1.01%, as listed in the top row of Table 4.1), produces errors on test material that lead to extra errors generated by the stress-assignment module, as compared to the stress-assignment module in the M-S-G system. In conclusion, it can be left to the stress-assignment module to implicitly learn to detect syllable boundaries necessary for producing correct stress assignments.

The utility of stress assignment

To investigate the utility of including stress assignments as input to grapheme-phoneme conversion, we perform an analysis analogous to investigating the utility of grapheme-phoneme conversion before stress assignment. We compare the results obtained on grapheme-phoneme conversion with the critical pair M-G-S and M-S-G. The accuracy difference is listed on the fifth row of Table 4.1. The difference indicates that when grapheme-phoneme conversion is preceded by stress assignment as in M-S-G, the accuracy of IGTREE on grapheme-phoneme conversion becomes significantly worse as compared to the case of M-G-S, where grapheme-phoneme conversion is the first module ($t(19) = 8.15$, $p < 0.001$). Thus, performing stress assignment before grapheme-phoneme conversion has a negative effect, compared to grapheme-phoneme conversion taking only takes letters and morphological boundaries as input.

4.5 Chapter conclusion

In Subsections 4.2.3 and 4.3.3 we have summarised the most important and significant comparisons of the investigated modular systems. We provide the conclusions drawn from these comparisons. The experiments were performed with two algorithms (IGTREE and BP), on a specific data set (the CELEX data) and thus our conclusions cannot be extrapolated to hold for other differently-constructed or differently-learned modular systems for English word pronunciation in general.

- Surprisingly, generalisation accuracy improves when the amount of modules is decreased from five to three. Graphemic parsing can best be integrated in grapheme-phoneme conversion, and syllabification can best be integrated in stress assignment. Thus, it is better to construct the three-module systems M-G-S and M-S-G than to construct the five-module systems M-A-G-Y-S and M-Y-S-A-G, respectively.

- With a five-module sequential architecture, it is better to perform graphemic parsing and grapheme-phoneme conversion (A-G) before syllabification and stress assignment (Y-S), than to perform Y-S before A-G. Similarly, with a three-module architecture, it is better to perform grapheme-phoneme conversion (G) before stress assignment (S) than vice versa.

- Including the output of modules as input to subsequent modules is profitable only in a limited number of cases in our experiments. Positive utility effects are found only with morphological segmentation as input to stress assign<ment, and grapheme-phoneme conversion as input to stress assignment (when trained with IGTREE under the adaptive variant).

- IGTREE performs consistently better than BP.

- Training modules under the adaptive variant yields consistently better generalisation accuracies than training modules under the perfect variant.

It can be asserted that sequential modular systems are subject to two opposite effects: first, a accuracy-increasing *summed utility* effect which allows separate modules to profit in certain cases from the output of previous modules, and second, a accuracy-decreasing *cascading error* effect that increases over

longer sequences of modules, and can only partly be countered by training modules under the adaptive variant. The system in which the summed-utility effect is maximal (when trained with IGTREE under the adaptive variant) is the three-module M-G-S system.

A comparison with word-pronunciation systems with different types of modular structure, in Chapters 5 and 6, will provide the opportunity to evaluate the best generalisation accuracy obtained with IGTREE on the M-G-S task.

Chapter 5

Modularisation by parallelising subtasks

The sequentiality of the modular systems investigated in Chapter 4 has been adopted from sequential modularisations of word pronunciation in two existing text-to-speech systems. The experiments reported in Chapter 4 demonstrate that inductively-learned modular systems yield reasonable generalisation accuracy. However, we could not determine the utility of sequential modularisation altogether, since for a sound comparison we lacked the generalisation accuracies of systems *without* sequential modularisation. Chapters 5 and 6 provide such results.

In this chapter we investigate modular systems in which subtasks are performed by one or more modules. Each module performs a subtask independent of the output of other modules (in one investigated system, there is only one module). All modules receive an identical input and generate different output *in parallel*. The output of each module constitutes a *partial* output of the word-pronunciation task; the output of the whole system is a synthesis of the partial outputs of its modules. A successful generalisation accuracy with parallelised subtasks would suggest a relative independence of the subtasks: in contrast with the utility assumptions underlying proposed sequential modular systems, the assumption underlying a parallel modular system is that the subtasks can best be performed without including the output of a module into the input of another module.

Modularisation by parallelisation implies defining a certain number n ($n \geq 1$) *partial subtasks*. A subtask is called partial when its output consists of sub-elements of the elements of the output of the word-pronunciation task,

which consists of phonemes with stress markers. We have investigated three systems in which three values of n are tested: (i) $n = 1$, (ii) $n = 2$, and (iii) $n = 26$:

ad (i) The system with $n = 1$ is the base-line case with a single module. In this system the word-pronunciation task is defined as mapping letter instances to phonemes and stress markers taken together in one combined class. The name of the system is GS, denoting that it performs grapheme-phoneme conversion (G) and stress assignment (S) together as a one-pass classification task. The GS system is described in Section 5.1. The algorithms applied to the GS task are BP, IB1, IB1-IG, and IGTREE[1].

ad (ii) In the system with $n = 2$, two subtasks are performed in parallel, viz. the mapping of letter windows to phonemes (G), and the mapping of letter windows to stress assignments (S). The name of the system is G/S, where the slash / denotes the parallel, independent processing of G and S. The G/S system is described in Section 5.2. All five algorithms are applied to the G/S task.

ad (iii) In the system with $n = 26$, the G subtask is divided into 25 partial subtasks, viz. the detection of 25 articulatory features (cf. Subsection 2.2.2; Appendix A). The 26th task is the stress-assignment (S) task as in the G/S system. Phonemes can be uniquely characterised by their articulatory feature values, hence, a successful detection of all features will yield a successful classification of the phoneme (Dietterich, Hild, and Bakiri, 1995; Dietterich and Bakiri, 1995; Wolters, 1997. This system is named ART/S, where ART stands for the 25 articulatory-feature-detection tasks. The ART/S system is described in Section 5.3. Due to the large number of experiments involved in learning the 25 partial subtasks, only IGTREE was employed because of its fast learning and low memory demands.

Section 5.4 provides a summary and a brief evaluation of the results obtained; a comparison is made between the accuracies of IGTREE on these three systems and the sequential-modular systems described in Chapter 4.

[1] C4.5 is left out of the comparison because it could not construct a tree when applied to the GS training material using less than 128 Mb of system memory.

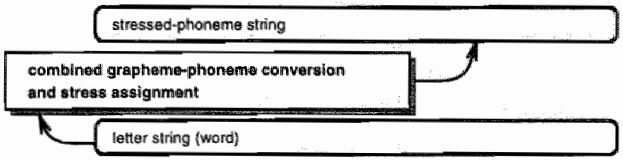


Figure 5.1: Visualisation of GS, a single-module word-pronunciation system. Round-edged boxes indicate input–output representations; the sharp-edged box denotes the single module. The input–output mappings performed by the module are depicted by arrows.

5.1 GS: Word pronunciation as a single one-pass task

GS is a single-module system in which only one classification task is performed in one pass. The GS task integrates grapheme-phoneme conversion and stress assignment: to classify letter windows as corresponding to a *phoneme with a stress marker* (henceforth referred to as PS). A PS can be either (i) a phoneme or a phonemic null (cf. Section 3.3) with stress marker '0', or (ii) a phoneme with stress marker '1' (i.e., the first phoneme of a syllable receiving primary stress), or (iii) a phoneme with stress marker '2' (i.e., the first phoneme of a syllable receiving secondary stress). Figure 5.1 visualises the simple architecture of GS which does not reflect any linguistic expert knowledge about decompositions of the word-pronunciation task. It only assumes the presence of letters at the input, and phonemes and stress markers at the output.

Table 5.1 displays example instances and their PS classifications generated on the basis of the word **booking** for the GS task. The phonemes with stress markers (PSs) are denoted by composite labels. For example, the first instance in Table 5.1, ___**book**, maps to class label /b/1, denoting a /b/ which is the first phoneme of a syllable receiving primary stress.

Symbolic-learning algorithms can only treat the PS classes as *atomic* classes, since the algorithms are restricted to learning one classification task at a time (but see Dietterich and Bakiri, 1995, who describe error-correcting output codes as a means for converting atomic classifications to strings of parallel binary classifications – an approach which is computationally not feasible for our purposes due to the very computing-intensive and memory-intensive modularity and learning). For BP, however, it is possible to learn more classification tasks simultaneously within the same MFN (Sejnowski and Rosenberg, 1987). Hence, when applying BP to GS, one is faced with a choice

instance number	left context			focus letter	right context			classification
1	_	_	_	**b**	**o**	**o**	**k**	/b/1
2	_	_	**b**	**o**	**o**	**k**	**i**	/u/0
3	_	**b**	**o**	**o**	**k**	**i**	**n**	/-/0
4	**b**	**o**	**o**	**k**	**i**	**n**	**g**	/k/0
5	**o**	**o**	**k**	**i**	**n**	**g**	_	/ɪ/0
6	**o**	**k**	**i**	**n**	**g**	_	_	/ŋ/0
7	**k**	**i**	**n**	**g**	_	_	_	/-/0

Table 5.1: Example of instances generated for task GS from the word **booking**.

how to represent combined classes: (i) by representing each subclass locally as a separate class (i.e., assigning each phoneme or stress marker its own uniquely-associated output unit, leading to $62 + 3 = 65$ output units); or (ii) by representing the combinations of subclasses locally (i.e., assigning one output unit to each possible combination of a phoneme and a stress marker, analogous to the method used with the symbolic-learning algorithms, leading to a maximum of $62 \times 3 = 186$ output units: actually, the number of combinations is 159). Although representations (i) and (ii) encode the same tasks, they constitute different learning tasks since they lead to MFNs with different network architectures. We present experimental results with BP using both representations to allow for comparisons between the alternatives. Henceforth, we refer to the experiments with the separately coded subclasses (i) as BP-SUB; the experiments with the combined classes (ii) are referred to as BP-COM.

GS: Experiments

From CELEX we constructed on the basis of the standard word base of 77,565 words with their corresponding phonemic transcription with stress markers, a data base containing 675,745 instances. The number of PS classes (i.e., all possible combinations of phonemes and stress markers) occurring in this data base of tasks is 159, which is fewer than the (Cartesian) product of the number of occurring subclasses ($62 \times 3 = 186$).

The algorithms applied to the GS task are BP in the variants BP-SUB and BP-COM, IB1, IB1-IG, and IGTREE. DC is also applied to the task to provide a baseline accuracy. Figure 5.2 displays all generalisation errors in terms of

RESULTS SUMMARY

best algorithm	IB1-IG
gen. error (%)	6.28
gen. error, words (%)	37.88

OVERLAP AND BIAS

overlap error (%)	20.10
class bias error (%)	81.16

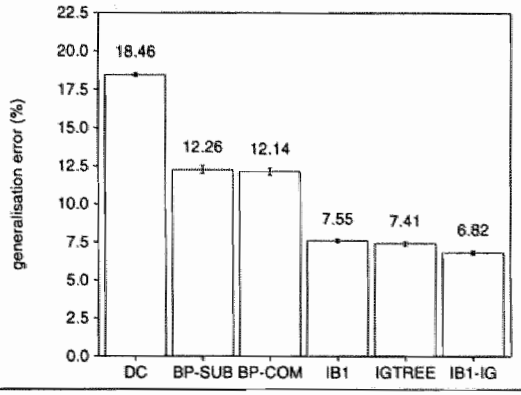


Figure 5.2: Results on the GS task. Results summary, bias and overlap errors (left), and generalisation errors in terms of the percentage incorrectly classified test instances of five algorithms (right).

incorrectly classified test instances, a results summary, and the overlap and bias errors. A test instance is classified incorrectly when the phoneme part is misclassified, or the stress-marker part, or both.

The results displayed in Figure 5.2 indicate that IB1-IG performs best on test instances. The differences between IB1-IG and the other algorithms are significant, the smallest difference being between IB1-IG and IGTREE ($t(19) = 9.05$, $p < 0.001$). There is no significant difference between the generalisation accuracies of BP-SUB and BP-COM: apparently, the difference in architecture does not lead to a different learning result. Encoding the output as locally-coded subclasses or as combined classes does not influence BP's generalisation accuracy (which is low compared to IGTREE, IB1, and IB1-IG).

On the basis of the classifications generated by the learned models it is also possible to compute the generalisation errors on phonemes separately. For example, if an instance is classified by an algorithm as /b/0, while the correct classification would be /b/1, the phoneme is classified correctly. By investigating only the phoneme part of the combined class a separate generalisation accuracy on correctly transcribed phonemes can be computed, and can be compared to the isolated counterpart grapheme-phoneme conversion subtask investigated in Section 3.3.

Figure 5.3 displays the results obtained with the five algorithms on the grapheme-phoneme-conversion part of the GS task (visualised in Figure 5.3 by the error bars labelled GS). The figure also gives the generalisation er-

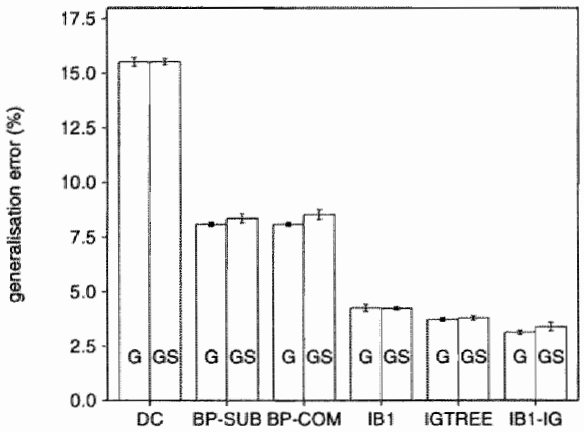


Figure 5.3: Generalisation errors in terms of the percentage incorrectly transcribed phonemes of five algorithms applied to the GS task, labelled GS, and to the isolated grapheme-phoneme conversion task, labelled G.

ror on phonemes obtained with the grapheme-phoneme conversion subtask described in Section 3.3 (the error bars labelled G). The results show that there is neither a consistent advantage nor a consistent disadvantage in learning grapheme-phoneme conversion as a partial subtask together with stress assignment, as compared to learning the task in isolation. Significant disadvantages are found with BP-SUB ($t(19) = 3.74$, $p < 0.001$), BP-COM ($t(19) = 5.85$, $p < 0.001$), and IB1-IG ($t(19) = 3.61$, $p < 0.01$). With DC, IGTREE and IB1, the differences are not significant.

The information-gain values of the features in the GS task are less differentiated than they are with the isolated grapheme-phoneme conversion task (cf. Appendix D), which may explain the significantly lower accuracy of IB1-IG on the grapheme-phoneme conversion subtask in the GS context, since IB1-IG works best with clearly differentiated information-gain values (Daelemans *et al.*, 1997a). Altogether it appears that performing the stress-assignment subtask alongside the grapheme-phoneme subtask does not help in attaining better accuracy on the latter task.

A comparison between the isolated subtask and the partial subtask within the GS task as done with grapheme-phoneme conversion cannot be done properly with the stress-assignment subtask and the stress-assignment part of the GS output, since the isolated subtask is performed on phonemes as

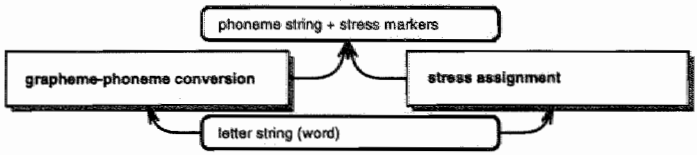


Figure 5.4: Visualisation of G/S, a word-pronunciation system containing two parallel modules. Round-edged boxes indicate input–output representations; sharp-edged boxes denote the modules. The input–output mappings performed by the modules are depicted by arrows.

input, whereas the stress-assignment part of the GS task is performed on letters as input. Notwithstanding the impossibility of a proper comparison we mention that the algorithms' accuracies on the stress-assignment part of the GS task are consistently lower than those of the same algorithms on the isolated subtask investigated in Section 3.5.

5.2 G/S: Performing two subtasks in parallel

A close alternative to the GS task is to perform the isolated subtasks of grapheme-phoneme conversion and stress assignment separately. It might be profitable to perform both tasks in parallel in a system with two independently-operating modules. This system, G/S, is investigated below. Figure 5.4 visualises the architecture of the system.

G/S: Experiments

The standard word base of 77,565 words with their phonemic transcription with stress markers is taken to create two data bases, one for each subtask of the G/S task, each data base containing 675,745 instances. The algorithms employed in the experiments are BP, IB1, IB1-IG, and IGTREE. For each algorithm, a double 10-fold CV experiment is performed on both grapheme-phoneme conversion and stress assignment. During each double 10-fold CV experiment the outputs of the two modules are joined for each test instance and the accuracy on correctly produced PSs of test words is computed. Figure 5.5 displays the generalisation errors, obtained with these algorithms on the G/S task. Also

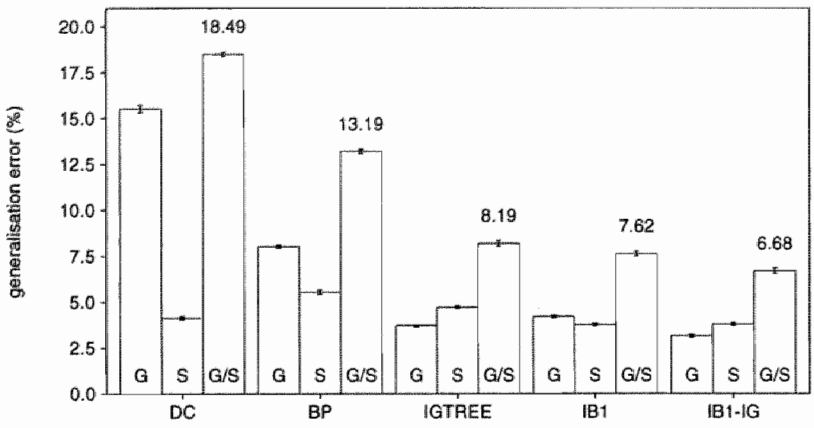


Figure 5.5: Results on the G/S task. Generalisation errors in terms of the percentage incorrectly classified PSs, of five algorithms applied to the isolated grapheme-phoneme conversion task (G), the stress-assignment task with letters as input (S), and the G/S task (G/S) (left).

displayed in Figure 5.5 are the errors on the two parallel subtasks measured separately.

We make four observations from Figure 5.5. First, on a general level, it displays once more the superiority of IB1 and IB1-IG over the other algorithms. IB1-IG performs significantly better than all other algorithms; the smallest difference between algorithms is between IB1 and IGTREE ($t(19) = 8.74, p < 0.001$).

Second, the generalisation errors of the algorithms on the G/S task are only slightly lower than the sum of the errors of the algorithms on the two parallel subtasks. In other words, when one of the two modules classifies an instance incorrectly, the other module classifies that instance correctly in most cases. A limited amount of instances is classified incorrectly by both algorithms, e.g., with IB1-IG not more than 3.6% of all misclassifications of PSs. This indicates that the two subtasks are inherently different.

Third, a mirrored symmetry can be seen in Figure 5.5 between the results of IGTREE and IB1. IGTREE's accuracy on isolated grapheme-phoneme conversion is better than that of IB1, while IB1's accuracy on stress assignment is better than that of IGTREE. This can be attributed to the difference between the feature information-gain values of both tasks, which are large for grapheme-phoneme

conversion, and relatively small for stress assignment (cf. Appendix D). For the stress assignment task, IGTREE fails to build trees fit for accurate generalisation, since they reflect a feature ordering that is not sufficiently clear (cf. Section 3.5); for the grapheme-phoneme conversion task, IB1 fails to recognise the fact that some letter positions are more important than others.

Fourth, low generalisation errors are produced by DC applied to the stress-assignment subtask with letters as input. They are significantly lower than those of IGTREE ($t(10) = 11.52, p < 0.001$) as well as of BP ($t(19) = 36.53, p < 0.001$). The bias of guessing class 0 (i.e., the absence of stress) on top of the overlap of 82.3% makes a relatively accurate guess for test instances (viz. 4.2% errors). The class 0 is indeed a strong majority class: 86.1% of all instances in the instance base are associated with it. Although stress assignment on the basis of letter windows is learned with the highest accuracy by IB1-IG (IB1 performing well too), the accuracy of DC shows that with less effort than with IB1-IG, only a slight (though significant) decrease in accuracy is obtained when the overlap between training and test instances is taken, and class '0' is produced whenever there is no overlap.

5.3 ART/S: Extending G/S with articulatory feature detection

ART/S incorporates the same parallelisation of word pronunciation into G and S; however, grapheme-phoneme conversion (G) is decomposed further into 25 partial subtasks. The combination of the 25 articulatory-feature-detection subtasks is referred to as ART. Each partial subtask detects the presence of one articulatory feature, with 7-letter instances as input. Since each phoneme is characterised by a unique articulatory-feature-value vector (Subsection 2.2.2), the output of the 25 partial subtasks can be synthesised to phoneme classifications. This parallelisation of partial subtasks may be profitable when it turns out that these subtasks are in sum easier and better learnable than the grapheme-phoneme conversion subtask G as a whole. Figure 5.6 visualises the architecture of the ART/S system.

The decomposition of the G subtask into articulatory-feature-detection subtasks is also described and analysed by Sejnowski and Rosenberg (1987) for their NETTALK model, Wolters (1997), and Dietterich *et al.* (1995). The latter work describes a comparison between ID3 (Quinlan, 1986) and BP, trained and tested on subsets of the NETTALK data (Sejnowski and Rosenberg, 1987). Dietterich *et al.* (1995) conclude that with certain extensions to ID3 and BP

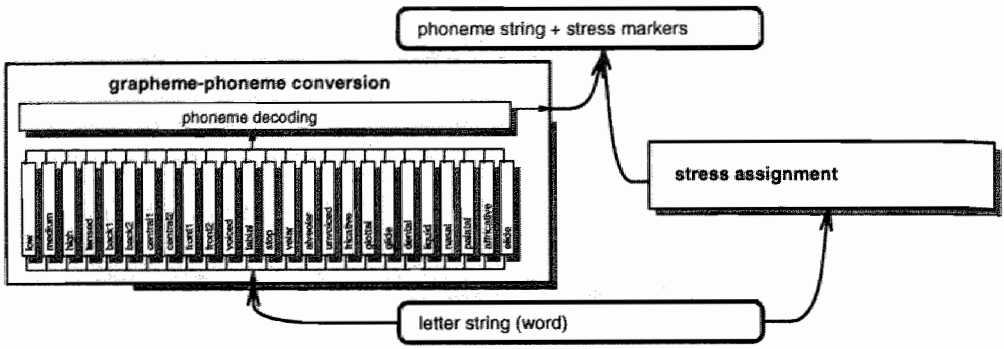


Figure 5.6: Visualisation of ART/S, a word-pronunciation system containing two parallel modules. Round-edged boxes indicate input–output representations; sharp-edged boxes denote the modules. The input–output mappings performed by the modules are depicted by arrows.

that make both algorithms more sensitive to statistical (i.e., frequency-based) properties of the data, the algorithms do not perform significantly different in terms of generalisation accuracy (Dietterich *et al.*, 1995).

In computational linguistics, articulatory features are used, e.g., as elements in morpho-phonological rules in the GRAFON-D system (Daelemans, 1987; Daelemans, 1988). Within phonological theories, many researchers (e.g., Halle, 1978; Clements and Hume, 1995) assume that articulatory features are the basic units of phonological representation. Thus, the basic assumption underlying the ART/S system is in accordance with these ideas.

The functioning of the ART component depends on the synthesis of the feature-detection outputs into phonemes. This synthesis is not trivial. Even though the 25 modules may produce feature-value vectors matching phoneme feature-value vectors perfectly, they may also produce vectors that have no phonemic counterpart. In those non-matching cases a phoneme label has to be searched of which the feature-value vector is in some way close to the vector produced by the 25 modules. This is done in the ART system by searching for a vector with a minimal Euclidean distance to the output vector. Euclidean distance, i.e., the similarity function also used in IB1, might be outperformed by a more complex matching component. This could, for instance, be another module (e.g., trained with IB1-IG) to map feature-value vectors to phonemes. However, we do not investigate such extra modularisation as it would ob-

instance number	left context			focus letter	right context			class
1	_	_	_	**b**	**o**	**o**	**k**	1
2	_	_	**b**	**o**	**o**	**k**	**i**	1
3	_	**b**	**o**	**o**	**k**	**i**	**n**	0
4	**b**	**o**	**o**	**k**	**i**	**n**	**g**	0
5	**o**	**o**	**k**	**i**	**n**	**g**	_	1
6	**o**	**k**	**i**	**n**	**g**	_	_	1
7	**k**	**i**	**n**	**g**	_	_	_	0

Table 5.2: Example of applying the windowing encoding scheme to the word **booking**, transcription /bukɪŋ/. The class labels encode the presence of the articulatory feature *voiced* in the respective phonemes.

fuscate seriously the distinction made between the systems investigated in Chapter 4 and in this chapter.

ART/S: Experiments

In Appendix A we have listed the articulatory-feature vectors of all consonant phonemes (Table A.4) and vowel phonemes (Table A.5) occurring in our data, and the names of the 25 articulatory features (Table A.3). We converted the standard word base of 77,565 words with their phonemic transcription with stress markers into 26 separate data bases (one for each articulatory feature, plus one data base for stress assignment), each containing 675,745 instances. For each articulatory-feature data set, instances are created which map to class 0 or 1. Class 0 means that the specific articulatory feature is not present in the articulatory-feature vector of the phonemic classification of the instance; class 1 indicates the presence of the articulatory feature in the articulatory-feature vector of the phonemic classification. Table 5.2 displays, as an example, the seven instances derived from the word **booking**, in which the class labels denote the absence or presence of the articulatory feature *voiced* (cf. Appendix A).

For each articulatory-feature-detection subtask and the stress-assignment subtask, a 10-fold CV experiment is performed with IGTREE (the stress-assignment subtask is the same subtask as performed with the G/S system). None of the other algorithms are applied to the ART/S subtasks due to the

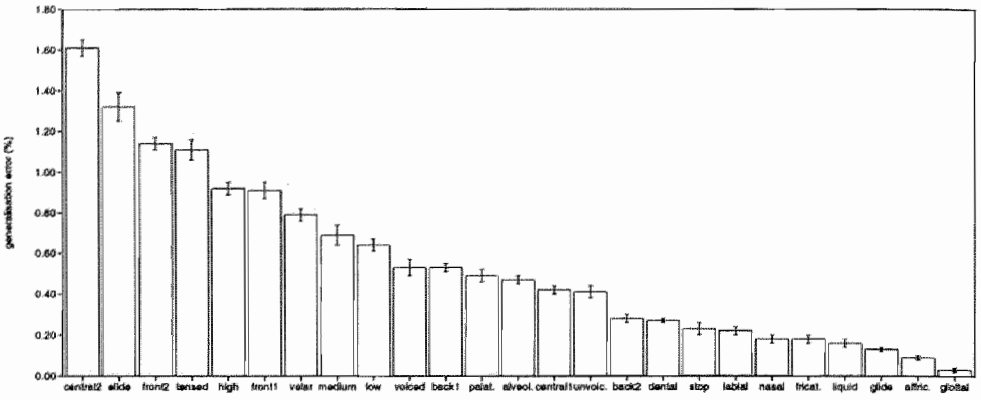


Figure 5.7: Generalisation errors in terms of the percentage incorrectly classified test instances by IGTREE applied to the 25 articulatory-feature-detection subtasks of ART/S.

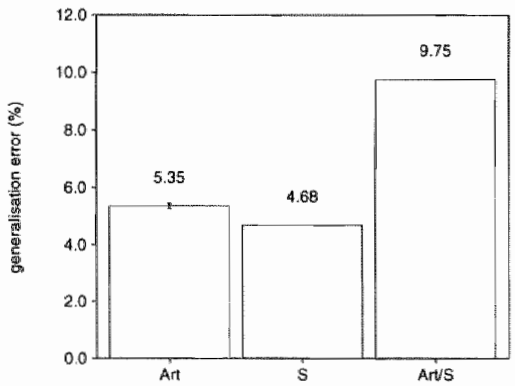


Figure 5.8: Generalisation accuracies of IGTREE applied to the isolated tasks of grapheme-phoneme conversion by articulatory-feature-detection (Art), stress assignment (S), and the ART/S task (Art/S).

large number of experiments and the relative speed of IGTREE's learning and classification. Figure 5.7 displays the average generalisation accuracies of IGTREE on all 25 articulatory-feature-detection subtasks.

All articulatory subtasks are performed with high generalisation accu-

racy. The feature detected with the lowest generalisation accuracy by IGTREE is *central2* (1.6% errors), which is a vowel feature (cf. Appendix A). The six articulatory features with the lowest generalisation accuracies are all vowel features, while the nine articulatory features with the highest generalisation accuracies are consonant features. These results indicate that classifying letter instances as mapping to consonant phonemes is generally easier than classifying instances as mapping to vowel phonemes. This reflects that in English, the pronunciation of vowels is more difficult in general than the pronunciation of consonants due to historical changes in articulation of particularly the English vowels (Kiparski, 1995).

Figure 5.8 displays the generalisation errors of IGTREE on phoneme classification by combining the articulatory-feature detector trees (labelled Art in the figure) on stress assignment (which is the same subtask as performed as part of the G/S system, labelled S in the figure), and on the combination of the two, the output of ART/S. As in Figure 5.5, the results in Figure 5.8 indicate that the generalisation error of the whole system (9.75%) is hardly less than the sum of the errors of its modules (10.03%). Only 2.9% of all incorrectly classified instances involve a false classification of both the phoneme and the stress marker.

5.4 Chapter conclusion

We have tested three word-pronunciation systems in which the task of converting letters to phonemes with stress markers is defined as performed by one, two, or 26 independent parallel-processing modules. All generalisation-accuracy results are summarised in Figure 5.9 in terms of generalisation errors. The best accuracy on parallelised word pronunciation is obtained with IB1-IG on the G/S task: 6.68% incorrectly classified PSs. This is narrowly but significantly better than the second-best overall accuracy, that of IB1-IG on the GS task, 6.82% incorrect PSs ($t(19) = 2.23, p < 0.05$). In both cases, the accuracy of IB1-IG is significantly better than that of all other algorithms, the smallest difference being between IB1-IG and IGTREE on the GS task ($t(19) = 9.05, p < 0.001$).

There are no major differences between the generalisation accuracies of the algorithms on the GS and the G/S tasks, although the two tasks appear to be very different. This fact is surprising, and is reflected by the small portion of instances for which both the phonemic and the stress classification is incorrect, in both systems, with all algorithms (e.g., 4.4% for the GS task and 3.6% for the G/S task with IB1-IG): since largely different instances appear to be

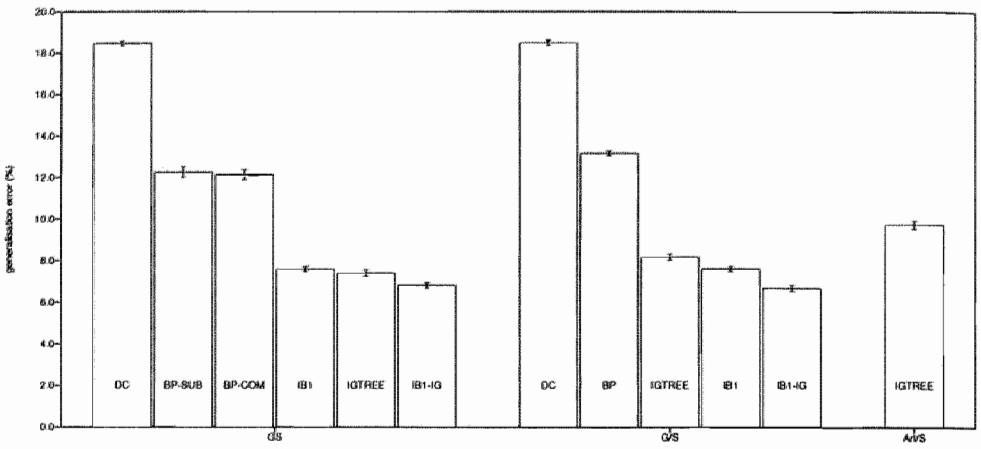


Figure 5.9: Generalisation errors of all algorithms applied to the GS, G/S, and ART/S tasks.

problematic for the two partial subtasks, it can be concluded that learning one of the subtasks is not helpful for learning the other subtask simultaneously.

The result obtained with IGTREE in the single experiment on the ART/S system signifies that the decomposition into 26 partial subtasks, among which 25 articulatory-feature-detection subtasks, is not profitable for generalisation accuracy. IGTREE's accuracy on GS as well as on G/S is significantly better ($t(19) = 29.79, p < 0.001$, and $t(19) = 19.86, p < 0.001$, respectively) than on ART/S. These results cannot be extrapolated to indicate that it is not profitable to perform parallel articulatory feature detection altogether; the accuracy of IB1-IG might well be better than IGTREE's (at immense memory costs, however).

The surplus in IGTREE's generalisation accuracy on GS over its accuracy of G/S and ART/S can be attributed largely to IGTREE's better generalisation accuracy on the stress-assignment subtask (4.0% errors) as compared to the parallelised stress-assignment subtask in G/S and ART/S (4.7% errors); there is no significant advantage in the generalisation accuracy on grapheme-phoneme conversion in GS (3.8%) as compared to the same subtask in G/S (3.7%).

Directly relevant to the goal of the present study, viz. to investigate whether inductive-learning algorithms can learn word-pronunciation even when linguistically-motivated decompositions of the task are avoided, is a comparison of GS, G/S, and ART/S to the four sequential-modular word-pronunciation systems investigated in Chapter 4. For this reason we compare

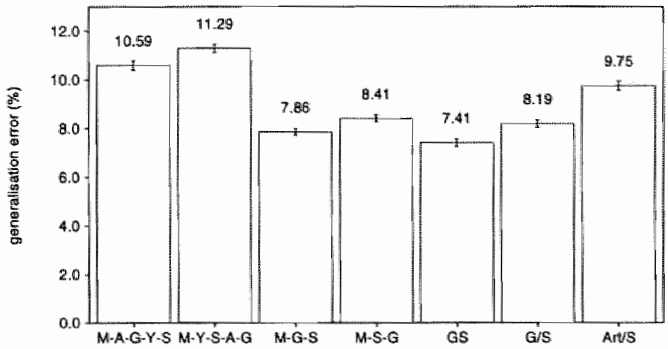


Figure 5.10: Generalisation errors of IGTREE on the M-A-G-Y-S, M-Y-S-A-G, M-G-S, M-S-G, GS, G/S, and ART/S systems.

the accuracies of IGTREE on these seven systems, IGTREE being the only algorithm applied to all of them. The comparison is visualised in Figure 5.10.

The first observation made from Figure 5.10 is that the lowest generalisation errors on word pronunciation obtained with IGTREE are with the GS system. IGTREE's accuracy on GS is significantly better than its accuracy on the M-G-S task ($t(19) = 6.90, p < 0.001$); all other differences have higher t-values. However, its accuracy on the G/S task, which is relatively low due to its low accuracy on the stress-assignment task, is significantly worse than that on the M-G-S task ($t(19) = 5.06, p < 0.001$), yet significantly better than all other accuracies (the closest difference being with the accuracy on the M-S-G task, $t(19) = 3.27, p < 0.01$).

The conclusion to be drawn from these results is that under our experimental conditions, IGTREE performs best on GS, the task which assumes no abstraction level at all and which represents the word-pronunciation task in the most simplified form tested: a direct mapping from letter instances to phonemes with stress markers (PSs).

Chapter 6

Modularisation by gating

The results of Chapters 4 and 5 suggest that decomposing word pronunciation into subtasks should be performed to a limited degree, if at all, to obtain good generalisation accuracy. The tested decompositions were all based on linguistic expert knowledge. Thus far we have not touched upon the possibility of decomposing word pronunciation into subtasks for which there is no particular linguistic motivation. The topic of this chapter is to investigate decompositions of the GS task of the latter type.

Decompositions of word pronunciation not based on linguistic motivations must be based on a criterion derived from the word-pronunciation data itself; there is no direct source of knowledge on task decompositions available otherwise. The data representing the GS task is a large set of associations between written words and their stressed pronunciations. Linguistic informedness allowed us earlier to decompose the word pronunciation task into partial subtasks, e.g., by splitting the output into phonemes and stress markers as in the G/S system (Section 5.2). A linguistically-uninformed decomposition of the GS task cannot involve splitting the output (phonemes with stress markers, PSs), hence involves finding a means to decompose the GS task on the basis of the input (the letter windows).

Such a purely input-based decomposition would constitute a *gating* system (Jacobs *et al.*, 1991) in which (i) the different decomposed modules accept the same type of input and produce the same type of output, i.e., they superficially perform the same task; and in which (ii) the modules deal with non-overlapping subsets of instances. The term gating refers to the extra component needed to control the presentation of input to one of the modules while blocking the presentation of that input to the other modules, the

so-called *gating module*. The task of the gating module is to decide for each input to which of the modules it is supposed to be presented. Thus, the gating module must be equipped with a *gating criterion* by which it can make these decisions.

Decomposing the input by a certain gating criterion leads to a partitioning of the original data set into subsets. It can be expected that constructing a system with an adequate gating component (i.e., a system outperforming GS) implies deriving a gating criterion from the data by which it can divide the data into subsets that each represent the word-pronunciation task in a manner that is better learnable for a learning algorithm (e.g., each subset contains instances that are inconsistent only with instances in the other subset; the degree of disjunctiveness of the classes is decreased, Holte *et al.*, 1989). When the subsets would not have this property, e.g., when the gating module would decide randomly, one might expect that the resulting gating system would perform worse than GS: a smaller subset of instances may have a negative effect on generalisation accuracy as the data becomes somewhat sparser. Empirical results are needed to confirm the expectation that a successful decomposition of the data by gating is only possible when the decomposed subsets are essentially different.

In this chapter we report on experiments in which three gating criteria are tested, each in one gating system. For the sake of comparability, all three gating criteria discern between two subsets.

1. **Randomised gating** is employed to test our expectation that a random partitioning of the data set will not lead to better learnable partial word-pronunciation tasks. It is also a test how well the word-pronunciation task can be learned when only half of the data is available. The system, called RND-GS, is described in Section 6.1.

2. **Typicality-based gating** discerns between a module trained and tested on instances that appear to be *typical* word-pronunciation instances, and a module trained and tested on the remaining instances. The notion of typicality is borrowed from Zhang (1992) (but see also Wilson, 1972, for earlier work in statistical pattern recognition), and the underlying idea is that a division of instances according to some measure of typicality or regularity (problematic as this is to determine without classification information) may lead to subsets that are essentially different. The system, called TYP-GS, is described in Section 6.2.

3. **Occurrence-based gating** divides the data on the basis of instance fre-

quency. One subset contains instances occurring below a certain occurrence threshold, and the other contains instances occurring at or above that threshold. On inspection it turns out that high-frequency instances often contain affixes and inflections, and that low-frequency instances generally contain morphologically simple words, e.g., singular nouns, noun stems, and verbs. The system, called OCC-GS, is described in Section 6.3.

The vast number of possible combinations of algorithms and (partial) tasks, combined with time limitations, led us to explore the three types of gating only with IB1, IB1-IG, and IGTREE, and not with BP (which learns too slowly) and C4.5 (which is not able to construct trees on the GS task within 128 Mb computer memory).

6.1 RND-GS: Randomised gating

The RND-GS system is constructed rather easily by implementing a gating module which decides randomly to which of the two GS modules an instance is presented as training or test material. The architecture, visualised in Figure 6.1, represents a double GS system, in which each of the modules is presented with about half of the training instances, and about half of the test instances. On average, both modules received about 304,000 training instances (instead of the average full training set of about 608,000 training instances), and about 33,500 test instances (instead of the usual 67,000). Thus far, we have not applied any algorithm on a data set of less than the maximum quantity provided by CELEX. Nevertheless, it has been claimed that with a corpus size of over 20,000 words (such as used in the NETTALK experiments, Sejnowski and Rosenberg, 1987), no significant improvements in generalisation accuracy can be expected to be obtained as compared to the results obtained with the 20,000-word corpus (Yvon, 1996). If it turns out that both modules of the RND-GS task perform significantly below the accuracy on the GS task (when trained with the same algorithm), we can argue that progress in generalisation accuracy can indeed be made when increasing the corpus size from 38,782 words (i.e., half the original CELEX word list) to 77,565 (i.e., the full original CELEX word list). Alternatively, when both partial GS tasks can be learned as accurately as the GS task itself, or even better, then our original assumption that the maximal amount of available data should be used in experiments with inductive-learning algorithms, would appear incorrect.

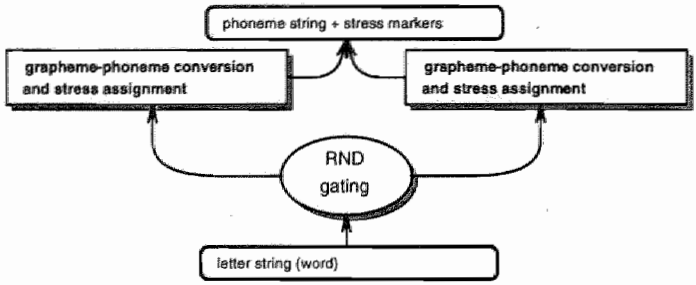


Figure 6.1: Visualisation of RND-GS, a word-pronunciation system containing a randomised-gating module (ellipse-shaped box) and two parallel GS modules (sharp-edged boxes). Round-edged boxes indicate input–output representations. The input–output mappings performed by the modules are depicted by arrows.

RND-GS: Experiments

The experiments on the RND-GS task, performed with IB1, IB1-IG, and IGTREE lead to consistently worse accuracy on the word-pronunciation task than that of the same algorithms on the GS task. The generalisation errors of the three algorithms are displayed in Figure 6.2, as well as the errors obtained by the same algorithms on the GS task. For all algorithms the difference between their accuracy on RND-GS is significantly worse than on GS (the smallest difference occurring with IB1-IG, $t(19) = 20.60$, $p < 0.001$). None of the differences between the generalisation accuracies on the partial subtasks and their joint accuracies are significant for any algorithm, which is what one would expect with approximately evenly-divided data sets of the same task.

The results show that the generalisation accuracy on word pronunciation of all three algorithms decreases significantly when the size of the lexicon is decreased from 77,565 to 38,782. Although a lexicon size of 38,782 word pronunciations might appear as containing sufficient word-pronunciation knowledge to be used in generalisation, there is still more information to be learned from the 77,565-word lexicon with our approach. An analysis of the overlap between training and test material with the 38,782-word data set reveals that the overlap between training and test set (i.e., the percentage test patterns which have a duplicate in the training set) is 73.6%, which is considerably lower than the 82.8% overlap in the 77,565-word data set (cf.

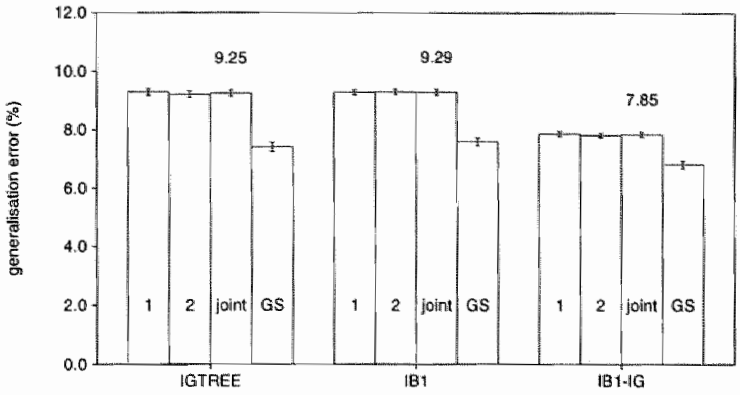


Figure 6.2: Generalisation errors (percentages of incorrectly classified phonemes with stress markers) of IGTREE, IB1, and IB1-IG, applied to RND-GS. Errors on the two randomly split subsets (labelled 1 and 2) are displayed, as well as the joint error of the whole RND-GS system (joint). Performance on the GS task is displayed for comparison (GS).

Subsection 2.4.2). Although a larger overlap may appear a trivial reason for better generalisation accuracy, it shows that learning word-pronunciation benefits from storing as much word-pronunciation instances as possible, because parts of words tend to reoccur in new, unseen words. Altogether, the results may be interpreted as being in accordance with our earlier claim that abstraction by data compression, i.e., forgetting, is harmful: as well as it is harmful to forget seemingly unnecessary information during learning, it is also harmful to throw away half the CELEX corpus. This might not hold for larger corpora, however; corpus sizes may be reached at which doubling the corpus size does not affect generalisation accuracy significantly.

6.2 TYP-GS: Typicality-based gating

To gate instances on the basis of their typicality, separating typical instances from the rest, a good definition of typicality is needed. We adopt a definition from Zhang (1992), who proposes a function to this end. Zhang computes typicalities of instances by taking both their feature values and their classifications into account (Zhang, 1992). He adopts the notions of *intra-concept*

similarity and *inter-concept similarity* (Rosch and Mervis, 1975) to do this. First, Zhang introduces a distance function similar to Equation 2.1 (p. 33), in which $W(f_i) = 1.0$ for all features (i.e., flat Euclidean distance), in which the distance between two instances X and Y is normalised by dividing the summed squared distance by n, the number of features, and in which $\delta(x_i, y_i)$ is given as Equation 2.2 (p. 33). The normalised distance function used by Zhang is given in Equation 6.1.

$$\Delta(X,Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(W(f_i)\delta(x_i,y_i))^2} \tag{6.1}$$

The intra-concept similarity of instance X with classification C is its similarity (i.e., the reverse of the distance) with all instances in the data set with the same classification C: this subset is referred to as X's *family*, $Fam(X)$. Equation 6.2 gives the intra-concept similarity function $Intra(X)$ ($|Fam(X)|$ being the number of instances in X's family, and $Fam(X)^i$ the ith instance in that family).

$$Intra(X) = \frac{1}{|Fam(X)|}\sum_{i=1}^{|Fam(X)|} 1.0 - \Delta(X, Fam(X)^i) \tag{6.2}$$

All remaining instances belong to the subset of unrelated instances, $Unr(X)$. The inter-concept similarity of an instance X, $Inter(X)$, is given in Equation 6.3 (with $|Unr(X)|$ being the number of instances unrelated to X, and $Unr(X)^i$ the ith instance in that subset).

$$Inter(X) = \frac{1}{|Unr(X)|}\sum_{i=1}^{|Unr(X)|} 1.0 - \Delta(X, Unr(X)^i) \tag{6.3}$$

The typicality of an instance X, $Typ(X)$, is the quotient of X's intra-concept similarity and X's inter-concept similarity, as given in Equation 6.4.

$$Typ(X) = \frac{Intra(X)}{Inter(X)} \tag{6.4}$$

An instance is typical when its intra-concept similarity is larger than its inter-concept similarity, which results in a typicality larger than 1. An instance is atypical when its intra-concept similarity is smaller than its inter-concept similarity, which results in a typicality between 0 and 1. Around typicality

value 1, instances cannot be sensibly called typical or atypical; Zhang (1992) refers to such instances as *boundary* instances.

A problem with the concept of typicality is that it cannot be applied to test instances, of which the classifications are not known. To handle both training and test instances with the same criterion the typicality-based gating module must apply a criterion based on the instances' feature values. The solution adopted here to circumvent this problem is to apply Zhang's (1992) typicality metric to the training instances (of which the classifications are known), and extract a set of the most typical instances (e.g., the 1,000 most typical instances, or the 10 most typical instances for each occurring classification). Given this small subset of the most typical instances, henceforth referred to as the *typical set*, the gating module can compare any instance, training or test, to the instances in the typical set and determine whether the instance is similar to one of the typical set's instances. If so, it is assumed that the instance is also typical, and is referred to the module handling typical instances. If not, the instance can be referred to the module handling all other instances.

The gating approach in TYP-GS is computationally expensive; the time complexity is $O(nf)$. It involves the computation of the typicality of all training instances, which is computationally of the same order as classification in IB1 and IB1-IG. After this first process TYP-GS becomes efficient: the typical set is extracted straightforwardly from the typicality data, after which the TYP-GS gating module can determine quickly to which of the two subsets an instance belongs. Unfortunately, the TYP-GS approach introduces a parameter determining the similarity threshold of new instances compared to the instances in the typical set. We chose to let the gating module perform an information-gain-weighted similarity match, and refer an instance to the module for typical instances when its similarity exceeds the sum of the information-gain-values of the five most important features of the GS TASK, viz. 4.50 (cf. Appendix D). Instances that match on less than five features with an instance in the typical set are not considered to be typical. At the same time the parameter value allows more than only duplicates of the few instances in the typical set to be considered typical; instances are allowed to mismatch slightly.

The architecture of the TYP-GS system is displayed in Figure 6.3.

TYP-GS: Experiments

To construct the TYP-GS system we start by creating a set of the most typical instances. First, we compute the typicality of all training instances. We then

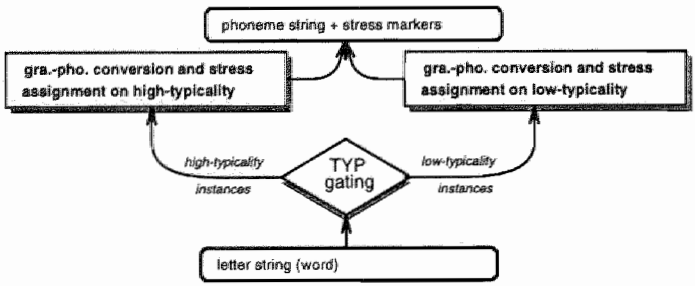


Figure 6.3: Visualisation of TYP-GS, a word-pronunciation system containing a gating module (diamond-shaped box) and two parallel GS modules (sharp-edged boxes). Round-edged boxes indicate input–output representations. The input–output mappings performed by the modules are depicted by arrows.

select for each of the 159 classes of the GS task the 10 most typical instances. This leads to a set of 1459 instances in the typical set[1]. Then, the gating module (TYP) is performed to split both the training set and the test set of each 10-fold CV experiment in two subsets. The predetermined parameter value 4.50 in the gating criterion leads to an average division of the original sets of instances to subsets of typical instances containing about 32%, and subsets of remaining instances containing about 68% of the original sets. Subsequently, IB1-IG, IB1, and IGTREE are employed to learn both partial GS tasks. The average generalisation errors of the three algorithms on the TYP-GS task, as well as their respective generalisation errors on GS, are displayed in Figure 6.4.

The most important result displayed by Figure 6.4 is that there is no negative effect nor a positive significant effect in gating the data according to the typicality criterion in the TYP module. None of the differences between the average generalisation errors of the three algorithms on the joint TYP-GS and the GS task are significant (the largest non-significant difference occurring with IB1, $t(19) = 1.43, p \geq 0.05$).

A somewhat counterintuitive result displayed in Figure 6.4 is that instances regarded as typical are classified with the lowest accuracy by all three algorithms. The best accuracies are obtained with the remaining subset which is supposed to contain the remaining, less typical instances. Apparently, the latter subset represents a part of the GS task that is easier to learn (i.e., with

[1] The amount of instances in the typical set is fewer than 10×159 since some classes occur fewer than 10 times.

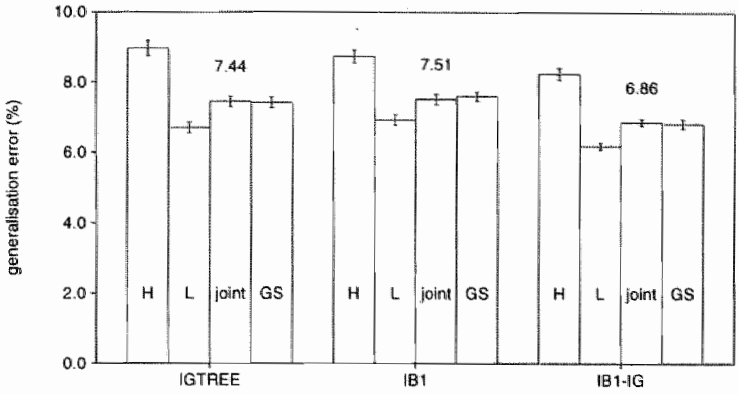


Figure 6.4: Generalisation errors (percentages of incorrectly classified phonemes with stress markers), of IGTREE, IB1, and IB1-IG, applied to TYP-GS. The errors on high-typicality subsets (H), low-typicality subsets (L), and joint error of the TYP-GS system (joint) are displayed. Performance on the GS task is displayed for comparison (GS).

better generalisation accuracy) than the part represented by the typical subset. For all three algorithms this difference is significant (the smallest difference occurring with IB1, $t(19) = 10.12, p < 0.001$). There appears to be more ambiguity in the subset of instances regarded as typical than in the subset of the remaining instances. This can be interpreted as a failure of the gating method to recognise truly typical instances. The gating method does allow instances to be slightly different than the instances in the typical set (i.e., two letters may differ). The seemingly small margin apparently introduces the relatively large amount of ambiguity. Thus, assuming that instances with a small distance to a typical instance are of the same class, appears to be wrong for a considerable number of such instances.

Despite the fact that the TYP-gating leads to one subset being learned better than the GS task as a whole, the accuracy of the three algorithms on the TYP-GS task is not significantly better than their accuracy on the GS task due to the counterweight of the error on the other subset.

6.3 OCC-GS: Occurrence-based gating

We have pointed out earlier, in Section 3.1, that a considerable amount of instances occur more than once in our data sets. This is not surprising, given the fact that many words are inflections or derivations of the same stem: e.g., the words **booking** and **bookings** share four identical instances with the four letters of the stem **book** in the focus position. Thus, stems such as **book**, or **work**, and many other noun stems, verb stems, and adjectives tend to occur more than once, at least twice (due to singular–plural pairs and verb inflections listed quite consistently in CELEX). Alternatively, inflections and derivations themselves occur in even more instances: e.g., the affix **able** (such as in **printable**, **readable**, **agreeable**, etc.) occurs quite often, leading to the instance **able**_ _ _ occurring 509 times in the full GS dataset.

The differences in occurrences between instances containing stems on the one hand and instances containing inflections and affixes on the other hand, is considerable, and is used here as the basis for gating. The gating criterion employs an occurrence threshold parameter determining to which of the two modules an instance must be referred. Our primary experiment employs a threshold parameter set at 50, i.e., an instance occurring fewer than 50 times in the training set is referred to the low-occurrence module, and an instance occurring 50 or more times is referred to the high-occurrence module. The value of 50 has a certain arbitrariness but appears to discern quite sharply instances with inflections and affixes from the rest of the instances. To illustrate this, Table 6.1 lists seven randomly-picked instances for each subset.

On inspection it turns out that the subset of instances occurring 50 times or more in the whole data set contains almost exclusively instances of the type displayed in the right column of Figure 6.1, viz. instances of which the focus position is occupied by a letter of a frequently-occurring inflection or affix such as **-es**, **-ing**, **-er**, **-able**, **ness**, **-ive**, **un-**, **be-**, and **anti-**.

The actual gating process for test instances is different from the gating process for training instances. The latter type of instances are gated simply on the basis of their occurrence within the training set; for test instances, however, the absolute threshold occurrence value of 50 cannot be applied sensibly since the test set is nine times as small as the training set in a 10-fold CV setup. Faced with the choice either to use an additional threshold value for test material derived from the value for training material (e.g., 7), or to check whether there are duplicates of the test instance in the training set and using the occurrence counts of the duplicate training instance, we choose the latter method. This means that the test set is split into a subset of instances that

example instances																	
in low-occurrence subset								in high-occurrence subset									
feature values							class	feature values									class
b	s	o	l	v	e	_	/l/0	_	_	_	a	n	t	i	/æ/2
_	c	l	i	q	u	e	/i/0	c	h	e	s	_	_	_	/z/0
n	t	e	r	t	a	i	/-/0	i	n	i	n	g	_	_	/-/0
n	e	l	e	g	a	n	/ɪ/0	a	l	i	z	e	_	_	/z/0
_	_	_	o	p	e	n	/əʊ/1	n	e	s	s	_	_	_	/-/0
v	e	n	g	e	d	_	/ʒ/0	t	i	o	n	_	_	_	/n/0
a	p	e	s	t	r	i	/s/0	r	i	e	r	_	_	_	/*/0

Table 6.1: Randomly-picked example instances in the low-occurrence subset (left, occurring fewer than 50 times) and in the high-occurrence subset (right, occurring 50 times or more).

either have no duplicate in the training set or occur fewer than 50 times in the training set, and a second subset of instances that have at least 50 duplicates in the training set.

The system created for testing occurrence-based gating is named OCC-GS. The architecture of the system is displayed in Figure 6.5, and again displays a gating module inserted before the two GS modules. Instances are gated through the gating module, are processed by one of the two modules, and are joined per word at the output side of the OCC-GS system.

OCC-GS: Experiments

The gating of the data by the OCC-gating module causes the high-occurrence training subset to contain about 11.0% percent of the total number of training instances, and the high-occurrence test subset to contain about 10.3% of the total number of test instances. This leads to 89.0% of the training instances and 89.7% of the test instances to be assigned to the low-occurrence module.

On the basis of Figure 6.6 we make three observations. First, none of the three algorithms can obtain a significantly better accuracy on OCC-GS than on GS (the largest non-significant difference occurs with IB1-IG, $t(19) = 1.44, p \geq 0.05$). Second, IGTREE's accuracy is significantly worse on the OCC-GS task than on the GS task ($t(19) = 26.18, p < 0.001$), due to a drastically low accuracy on the partial subtask on high-occurrence instances. An explanation for the latter low accuracy is that the information-gain values of the features of the low-

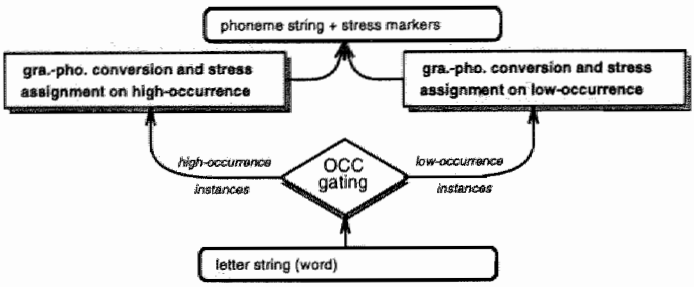


Figure 6.5: Visualisation of OCC-GS, a word-pronunciation system containing a gating module (diamond-shaped box) and two parallel GS modules (sharp-edged boxes). Round-edged boxes indicate input–output representations. The input–output mappings performed by the modules are depicted by arrows.

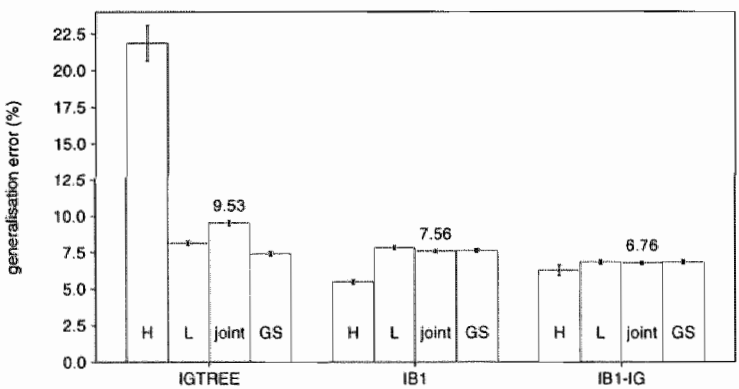


Figure 6.6: Generalisation errors in terms of the percentage incorrectly classified phonemes with stress markers, of IGTREE, IB1, and IB1-IG, applied to OCC-GS. Displayed are the errors on high-occurrence subsets (H), low-occurrence subsets (L), and joint error of the whole OCC-GS system (joint).

occurrence instances are relatively similar, which is a notorious cause of IGTREE to perform badly on test instances. Third, the average generalisation error of IB1 on the high-occurrence subset is very low (viz. 5.48% incorrectly-classified PSs), which is a remarkable contrast to the high error generated by IGTREE on the same subset. This contrast can be explained as follows: the test set of high-occurrence instances fully overlaps with the training set, which implies that lookup through the full instance base of the most probable classification will always produce a perfect match (unless there is a data-inherent class ambiguity between two identical instances). IB1 and IB1-IG are at an advantage when retrieving classifications by lookup, since they have stored all training instances in memory. In contrast, the retrieval of knowledge from the tree in IGTREE is guided by a badly-motivated ordering of features: e.g., computing the information-gain values of the features of the high-occurrence instances it turns out that the second-highest information-gain-value belongs to the third letter on the right from the focus. Mismatches on this feature cause IGTREE to halt retrieval and produce a best-guess classification from depth two of the tree: the results indicate that these best guesses are rather inaccurate.

6.4 Chapter conclusion

Figure 6.7 summarises the generalisation errors obtained with IB1, IB1-IG, and IGTREE on the three gating systems RND-GS, TYP-GS, and OCC-GS, and repeats, for comparison, the generalisation errors of the algorithms on the GS task.

An explanation for the low accuracy of all algorithms on the randomised-gating system RND-GS is that the data is made sparser when divided over two modules. Just as it is harmful to store not all information contained in training instances when learning the task, it is also harmful to generalisation accuracy to forget half of the data. In a lexicon of 77,565 words, more information on word-pronunciation appears to be present and is actually learned by inductive-learning algorithms than in a lexicon of half the size.

Analysing the results obtained with the typicality-based-gating system TYP-GS, it appears that estimating typicality of an instance by matching that instance to a small set of very typical instances is a rough method that nevertheless leads to a viable decomposition. The same can be said of occurrence-based gating in OCC-GS. However, the gating in both systems does not lead to significant improvements over the accuracies on the GS task.

In general, we can conclude from the results that it is possible to construct gating systems on the basis of data-oriented criteria without losing general-

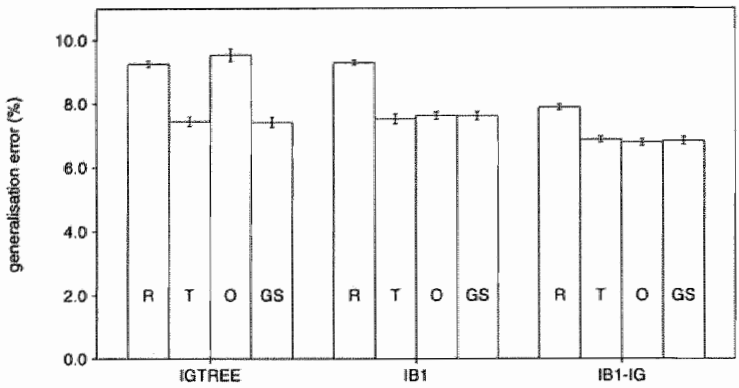


Figure 6.7: Generalisation errors in terms of the percentage incorrectly classified phonemes with stress markers, of IGTREE, IB1, and IB1-IG, applied to RND-GS (R), TYP-GS (T), OCC-GS (O), and, for comparison, GS (GS).

isation accuracy. The estimated typicality of an instance (a rather complex measurement) and the occurrence of an instance (a simple measurement) both provide valid gating criteria: therefore, it can be argued that both relate to inherent features of word-pronunciation. First, instances can be roughly classified as typical or not typical. However, since results pointed out that more generalisation errors are made on instances considered typical, as on instances considered not typical, we refrain from claiming that Zhang's (1992) metric can successfully be employed to make a sharp distinction between 'typical' and 'not typical' with instances of the word-pronunciation task. Second, counting an instance's occurrence in a training set indicates that it is likely that the instance incorporates a high-frequent inflection or affix. No exact knowledge of the position of morpheme boundaries can be inferred from this indication, but is demonstrated to be employable in a well-performing gating system. Thus, clues on the morphological structure of words can be discovered in the data straightforwardly. On the other hand it is just as viable to use the undecomposed data for learning word pronunciation. This conclusion applies to processing on serial computers, the default case for our study; on parallel processing computers, automatic decomposition by gating would be the favourable option (cf. Chan and Stolfo, 1995).

Chapter 7

Discussion

The empirical investigations described in Chapters 3 to 6 have produced a large number of results, most of which have been analysed in the context of the appropriate chapter's topic. The first goal of this chapter is to summarise and discuss the results globally in relation to the problem statement, i.e., whether inductive-learning algorithms can learn to pronounce words with adequate generalisation accuracy, even when the task decomposition reflects none of the abstraction levels assumed necessary by linguistic experts. In Section 7.1 we attempt to do this by measuring and comparing what the problem statement refers to, i.e., generalisation accuracy. Moreover, adding to the comparative analyses of generalisation accuracies, we provide details on memory usage and processing speed to give a computationally balanced view of the strengths and weaknesses of the algorithms. Although these additional analyses are besides the focal point of answering the problem statement, they do provide insight into the feasibility and the applicability of the general approach. In Section 7.2 the analyses from Section 7.1 are used for proposing an assembled hybrid word-pronunciation system which aims at combining some profitable features from the word-pronunciation systems investigated thus far in order to optimise generalisation accuracy.

In Section 7.3 we discuss and analyse *why* lazy learning (IB1 and IB1-IG) is the best suited learning method for learning word pronunciation for the data set under investigation and under our experimental settings, and why abstraction by forgetting during learning is harmful. These questions are intimately related to the question which are the characteristics of the word-pronunication task as we formulate it that make it so amenable to lazy learning.

Sections 7.4 and 7.5 discuss the contributions, limitations, and implications of the present study. In Section 7.4, the relations are described between our approach and work in related areas: the interdisciplinary machine learning of natural language (MLNL), and computational morphology and phonology. The subsequent Section 7.5 identifies some of the limitations of the present approach and analyses how these could be overcome. Indications are given of potential expansions and future work.

7.1 Comparing inductively-learned word-pronunciation systems

To answer the problem statement we must demonstrate that the inductive-learning algorithms can learn word pronunciations adequately. In addition, we must demonstrate that this performance remains adequate even when the system does not reflect linguistic expert assumptions about abstraction levels. For this demonstration we establish (i) a lower-bound generalisation accuracy threshold for considering a generalisation accuracy adequate, (ii) an upper-bound accuracy threshold determined by the inherent ambiguity in the data; earlier, we have described (iii) which of our word-pronunciation systems does not reflect any mainstream linguistic expert knowledge on abstraction levels.

First, we establish a sound lower-bound threshold for considering a generalisation accuracy adequate. This is not trivial. As Yvon (1996) notes, high-quality text-to-speech synthesis demands that 80%–90% of the words are pronounced flawlessly. A word's pronunciation is flawless unless one or more of its phonemes is incorrect[1]. Thus, given an average word length of 8 letters in our data, and the worst-case assumption that erroneously-classified phonemes are uniformly distributed over words, only about $(20/8) = 2.5\%$ classification errors on phonemes can be tolerated. In addition, it can be concluded from the overviews of machine-learned models of word pronunciations provided by Yvon (1996) that reported accuracies on words claimed to be adequate range from about 50% to about 67% flawless word transcriptions (e.g., Dietterich and Bakiri, 1991; Golding, 1991), and from about 90% to about 95% correctly classified phonemes (e.g., Dietterich and Bakiri, 1991; Yvon, 1996). Given these fuzzy boundaries (obtained with statistically incomparable data sets) we propose to consider a generalisation accuracy on phonemes

[1] Yvon's (1996) accuracy threshold estimate does not include stress markers; it focuses on phonemes which are of more importance in text-to-speech systems than word stress.

of 5% or lower as adequate. This number does not refer to scores on *PSs* (phonemes with stress markers), which has been our primary focus. (Most related work indeed refers to accuracies on test phonemes; only Dietterich *et al.* (1995) report on joint accuracies on phonemes and stress markers, viz. of ID3 (Quinlan, 1986) and BP applied to the NETTALK data; ID3 classifies 65.2% PSs correctly, while BP yields 71.3% correctly classified PSs (Dietterich *et al.*, 1995). These accuracies are considerably lower than ours, yet the NETTALK data is hard to compare to the CELEX data. In the original NETTALK study (Sejnowski and Rosenberg, 1987) both phonemes and stress markers are encoded in the network's output, but no accuracies on the combined classifications are reported. Stanfill and Waltz (1986), describing their MBRTALK system applied to a subset of the NETTALK data, report on phonemes (86% correct test phonemes) and words (47% correctly pronounced test words). Yvon (1996) explicitly splits the two NETTALK subtasks and does not integrate their output.) In order to compare our generalisation errors with the 5% threshold we need to include the generalisation accuracies on phonemes of our word-pronunciation systems in our analysis.

Second, we determine the upper-bound accuracy threshold. The word-pronunciation data as used in our experiments contains two types of classification ambiguity. The first type of ambiguity stems from homographs such as **read**, **object**, and **suspect**, and cannot be learned successfully since no knowledge is available to determine which of the two pronunciations is appropriate when the word is pronounced in isolation. The second type of ambiguity stems from the limited context captured by the letter windows (viz. three left-context letters and three right-context letters). This blocks the disambiguation of the pronunciation of homograph pairs such as **photograph** / **photography** and **allergy** / **allergic**, in which the pronunciation of the first vowel (**o** and **a**, respectively) is determined by a letter difference occurring more than three characters to the right. Because of both types of ambiguities, the word-pronunciation task as represented by our data cannot be learned with 0% generalisation errors. A sensible approximation of the theoretical upper-bound accuracy on the word-pronunciation task can be determined by training an algorithm on the full data set of word-pronunciation and testing its *reproduction accuracy* on this full data set. We have performed this experiment by training and testing IB1-IG on the full data sets of (i) the isolated grapheme-phoneme conversion subtask (cf. 3.3), (ii) the isolated stress-assignment subtask (cf. 3.5), and (iii) the GS task. We have joined the outputs of experiments (i) and (ii) to obtain an estimate of the upper-bound accuracy

task	% incorrect training instances	phonemes	stress	words
G/S	3.88	1.30	2.73	24.96
GS	3.86	1.31	2.73	24.82

Table 7.1: Reproduction (upper bound) accuracy, in terms of percentage incorrectly classified training instances, phonemes, stress assignments, and words, of IB1-IG trained and tested on the full data sets of the G/S task (i.e., the isolated grapheme-phoneme and stress-assignment subtasks) and of the GS task.

on the G/S task. The upper-bound results on the G/S and GS tasks are listed in Table 7.1. There is no significant difference in the accuracies of IB1-IG on both tasks. We adopt the percentage errors made on phonemes, which is at best 1.30%, as the estimated theoretical upper-bound accuracy. Thus, we consider an algorithm's accuracy to be adequate when it produces less than 5% errors on phonemes; an accuracy close to 1.30% would be considered excellent.

Third, we establish again which of our systems does not reflect in its architecture any mainstream linguistic expert knowledge on abstraction levels. The experiments with the different word-pronunciation systems in Chapters 4 to 6 were described in an order reflecting roughly a decreasing amount of incorporated linguistic expert knowledge. In two systems, viz. M-A-G-Y-S and M-Y-S-A-G, linguistic expert knowledge is employed to determine both the decomposition (i.e., the identification of the different subtasks) and the modular structure (i.e., sequential). In contrast, in the four systems GS, RND-GS, TYP-GS, and OCC-GS, neither the decomposition nor the modular structure is adopted from linguistic expert knowledge. The remaining four systems, viz. M-G-S, M-S-G, G/S, and ART/S, reflect a mixture of linguistic and non-linguistic (empirical) motivations. Table 7.2 displays the generalisation accuracies obtained on all word-pronunciation systems in terms of their generalisation errors on misclassified phonemes as well as on misclassified PSs. The systems are grouped according to their respective amount of incorporated linguistic expert knowledge; the accuracies on the M-A-G-Y-S and M-Y-S-A-G (both under the adaptive variant) are listed at the top of Table 7.2 and the GS and gating variants on GS are listed at the bottom.

The level of adequate accuracy, i.e., a generalisation error below 5% incorrectly classified test phonemes, is obtained with IGTREE, IB1, and IB1-IG,

system	classification type	algorithm DC	BP	IGTREE	IB1	IB1-IG
M-A-G-Y-S	phonemes	–	6.80	6.34	–	–
	PSs	–	13.58	10.59	–	–
M-Y-S-A-G	phonemes	–	7.89	6.78	–	–
	PSs	–	14.26	11.29	–	–
M-G-S	phonemes	–	6.70	3.99	–	–
	PSs	–	12.92	7.86	–	–
M-S-G	phonemes	–	7.60	5.43	–	–
	PSs	–	13.56	9.27	–	–
G/S	phonemes	15.52	8.10	3.72	3.76	3.14
	PSs	18.49	13.19	8.19	7.62	6.68
ART/S	phonemes	–	–	5.44	–	–
	PSs	–	–	9.75	–	–
GS	phonemes	15.52	7.65	3.79	4.23	3.39
	PSs	18.46	12.14	7.41	7.60	6.82
RND-GS	phonemes	–	–	5.29	5.65	4.98
	PSs	–	–	9.26	9.29	7.85
TYP-GS	phonemes	–	–	3.82	4.13	3.34
	PSs	–	–	7.44	7.51	6.86
OCC-GS	phonemes	–	–	5.93	4.18	3.20
	PSs	–	–	9.53	7.56	6.76

Table 7.2: Summary of the generalisation errors on phonemes as well as PSs (phonemes with stress markers), obtained with all word-pronunciation systems with DC, BP, IGTREE, IB1, and IB1-IG. Generalisation errors on phonemes considered adequate (i.e., lower than 5%) are boxed. When an algorithm has not been applied to a system, the corresponding cell contains '–'.

on the G/S task and on the GS task. Moreover, IB1 and IB1-IG perform adequately on the TYP-GS and OCC-GS tasks, IB1 performs adequately on the RND-GS task, and IGTREE performs adequately on the M-G-S task (adaptive variant). In terms of incorrectly classified test phonemes the lowest error is obtained with IB1-IG on the G/S task: 3.14%, i.e., 96.86% correctly classified phonemes. This is significantly better than IB1-IG's accuracy on phonemes on the GS task ($t(19) = 3.61, p < 0.01$), but is not significantly different from the accuracy of IB1-IG on OCC-GS ($t(19) = 1.58, p \geq 0.05$). In terms of incorrectly classified PSs the lowest error is yet again obtained with IB1-IG on the G/S task: 6.68%, i.e., 93.32% correctly classified PSs. This accuracy is narrowly but significantly better than the accuracy of IB1-IG on the GS task ($t(19) = 2.23, p < 0.05$), but not significantly better than IB1-IG's accuracy on the OCC-GS task ($t(19) = 1.36, p \geq 0.05$). In other words, the best adequate accuracy is obtained with IB1-IG on the G/S task and the OCC-GS task. The difference between this accuracy (3.14% incorrect phonemes) and the theoretical upper bound (1.30% incorrect phonemes) suggests that improvement might be possible (we continue discussing the issue of accuracy improvement in Section 7.5).

In sum, the results indicate that the word-pronunciation task can be learned adequately by the two instance-based learning algorithms IB1 and IB1-IG and the decision-tree learning algorithm IGTREE, when it is decomposed into three subtasks performed in sequence (in the M-G-S system), when it is decomposed in two subtasks performed in parallel (in the G/S system), and when it is not decomposed at all (in the GS task). The latter result indicates that IB1, IB1-IG, and IGTREE can learn to pronounce words with adequate generalisation accuracy even when the system architecture reflects none of the abstraction levels assumed necessary by linguistic experts.

The results call for an analysis of the results obtained with BP and C4.5. For the case of BP it can be claimed that the tested network architectures in combination with the learning parameters fall short in learning the word-pronunciation task. Our fixed experimental settings have not permitted us to vary systematically parameters that might have had an influence on generalisation accuracy, e.g., the number of hidden units (fixed at 50 throughout all experiments). Analysing the performance results of BP, it cannot be concluded that it is impossible for BP to learn word-pronunciation to an adequate level. It appears that the presently-tested network architecture, trained under the parameter values as described in Subsection 2.4.3, cannot represent the word-pronunciation task adequately. Whether this is due to the lack of space (i.e.,

exp. #	learning rate	momentum	patience threshold	# hidden units	% incorrect PSs
1	0.100	0.4	0.025	50	12.58
2	**0.050**	0.4	0.025	50	12.70
3	**0.025**	0.4	0.025	50	12.35
4	0.100	**0.2**	0.025	50	12.54
5	0.100	**0.8**	0.025	50	12.95
6	0.100	0.4	**0.010**	50	12.20
7	0.100	0.4	0.025	**100**	11.85
8	0.100	0.4	0.025	**200**	10.04
9	**0.025**	**0.2**	**0.010**	**200**	9.89
10	0.100	**0.2**	**0.010**	**200**	9.68

Table 7.3: Percentages of incorrectly classified phonemes with stress markers (PSs), computed on the first partitioning of the GS data, for ten different parameter settings of BP trained on the GS task. Each non-default parameter value is printed in bold. The top line denotes the experiment with the default parameter values.

too few hidden units) or the inability to represent the data due to its specific characteristics (e.g., sparseness or disjunctiveness of classes; cf. Section 7.3) is unclear without further empirical investigations. Table 7.3 displays the results obtained with performing explorative experiments on a single 10-fold CV partitioning of the GS data, varying the learning rate, the momentum,, the stopping criterion's patience threshold, and the number of hidden units (cf. Subsection 2.1.2).

The table shows that marked improvements in generalisation accuracy can be obtained with increased numbers of hidden units. Experiments 7 and 8, in which the number of hidden units is increased to 100 and 200, respectively, yield markedly lower errors than the default-setting experiment 1 and the other experiments 2 to 6. Changing learning rate, momentum, and the stopping criterion's patience threshold value affect generalisation accuracy to a small degree. The best accuracy is obtained with leaving the learning rate at 0.100, setting the momentum at 0.2, setting the patience threshold to 0.010, and setting the number of hidden units to 200 (experiment 10). Nevertheless,

an error of 9.86% incorrectly classified PSs is still quite high compared to the average percentages of errors made by IGTREE (7.41%), IB1 (7.60%), and IB1-IG (6.82%). At the cost of very long training phases[2], increases in hidden units may lead to further improvements in generalisation accuracy. The fact that our study is performed with non-optimal parameter settings (which allowed us to perform the experiments in the given time) is a combined weakness of our approach in which parameters are set once for many tasks, and of BP for having a large amount of parameters, some of which have considerable impact on the accuracy of BP on the GS task as shown. Further empirical investigations are needed to determine whether algorithms such as Quickprop (Fahlman, 1988) or Cascade-correlation (Fahlman and Lebière, 1990) may aid in optimising generalisation accuracy or processing speed.

The C4.5 algorithm has not been applied to the GS task as it failed to build a decision tree within the practical limitations of 128 Mb of computer memory. C4.5's implementation[3] incorporate some computational inefficiencies (Daelemans *et al.*, 1997a) that could be avoided without affecting the performance of the algorithm. Hence, the results obtained with C4.5 should be interpreted as a general failure of C4.5 to deal with the word-pronunciation task, and indicate that a more efficient, less memory-consuming implementation of C4.5 would be appropriate.

7.1.1 Additional comparisons between algorithms

While the problem statement only mentions generalisation accuracy as the criterion to measure success in reaching the goal of learning word pronunciation adequately, it is relevant from a computational point of view to compare the algorithms on other features as well. This section presents three additional analyses of the algorithms: (i) the asymptotic complexities of the algorithms as regards classifier construction, classifier storage, and instance classification; (ii) their actual memory requirements on the GS task, and (ii) the time needed to learn the GS task and to classify instances of that task.

[2] Experiment 10 took 12 days to converge on a SUN Sparc Classic, which is very long for a single-partition experiment.

[3] These statements refer to release 7 of C4.5, but apply also to release 8. However, a new commercial version of C4.5, C5.0, is able, by certain memory optimisations, to learn the GS task with the same parameters as investigated in this thesis. C5.0's generalisation error on test instances is 7.53%, which is slightly but significantly worse than IGTREE ($t(19) = 1.73, p < 0.05$); the average tree built by C5.0 contains 25,663 nodes.

algorithm	classifier construction	classifier storage	classification
BP	$O(n\ f\ h\ c)$	$O(f\ h\ c)$	$O(f\ h\ c)$
IB1	$O(n)$	$O(n\ f)$	$O(n\ f)$
IB1-IG	$O(n)$	$O(n\ f)$	$O(n\ f)$
IGTREE	$O(n\ lg(v)\ f)$	$O(n)$	$O(f\ lg(v))$
C4.5	$O(n\ lg(v)\ f)$	$O(n)$	$O(f\ lg(v))$

Table 7.4: Asymptotic complexities (in $O()$ notation) of classifier construction, classifier storage, and classification of BP, IB1, IB1-IG, IGTREE, and C4.5.

Asymptotic complexity

An analysis of the asymptotic complexities of BP, IB1, IB1-IG, IGTREE, and C4.5 as employed in our study provides an indication of the differences in the worst-case performance of the five algorithms, before we turn to their actual performance in terms of memory requirements and processing speed. In Table 7.4 we have gathered for each of the five algorithms the order of complexity of (i) classifier construction (i.e., the computational processing cost of training an MFN with one hidden layer, constructing an instance base, or inducing a decision tree), (ii) classifier storage (i.e., the memory cost of storing a trained MFN, an instance base, or an induced decision tree), and (iii) classification (i.e., the computational processing cost of classifying instances) (cf. Daelemans *et al.*, 1997a). In the table, n denotes the number of instances in the training set; f denotes the number of features; v denotes the average branching factor of IGTREE and C4.5 (i.e., the average number of values represented at arcs stemming from a node); c represents the number of classes; and h represents the number of hidden units in the MFN trained with BP. For our data, n is very large; when processing or storage is dependent of n, it can be said to be less favourable than processing or storage independent of n for our data.

Classifier construction is particularly favourable in complexity for IB1 and IB1-IG, which only need to store n instances in memory. IGTREE and C4.5 both need $O(n\ lg(v)\ f)$ when values are alphabetically sorted (so that binary search can be implemented). Classifier construction in BP is less favourable, since it is not only dependent of n and f but also of h and c. Complexity analyses

of classifier storage show, in reverse, that BP is not dependent of n, while the other four algorithms are. IGTREE and C4.5 need to store, in the worst case, n (the maximal number of leaves) $+((n-1)*(v-1))$ (the maximal number of non-terminal nodes), hence $O(n)$. IB1 and IB1-IG store for all n all feature-value information; hence $O(n\,f)$. Classification of instances is dependent of n in IB1 and IB1-IG; instance classification in BP, IGTREE, and C4.5 is independent of n and therefore favourable (for our data).

Memory requirements

Table 7.5 lists the average memory requirements of all tested algorithms on all reported word-pronunciation systems, as well as on the isolated subtasks reported in Chapter 3. The results in the Table clearly indicate the magnitude of the difference, for all systems, between the memory requirements of the instance-based (lazy) learning algorithms IB1 and IB1-IG (and DC) on the one hand, and those of the two decision-tree learning algorithms IGTREE and C4.5, and the connectionist BP algorithm on the other hand. First, BP's modest memory requirements contrast strongly with the other algorithms' requirements, especially those of DC, considering that for most tasks BP performs significantly better than DC. Second, the decision-tree-learning algorithms IGTREE and C4.5 require comparable amounts of memory, except for the tasks involving grapheme-phoneme conversion; in the latter cases C4.5 failed to induce trees within 128 Mb working memory. Third, the lazy-learning algorithms require the full training sets to reside in memory, which amounts to 4,751 Kb per module.

The results suggest that adequate accuracy may be reached with a limited amount of compression during learning; when compression is too high, generalisation accuracy becomes lower than adequate. For example, the network used by BP to learn the GS task occupies 97.8% less memory than the instance bases used by IB1 and IB1-IG. The decision tree constructed by IGTREE occupies 84.0% less memory. The amount of compression by BP leads to significantly worse generalisation accuracy, falling below the level of adequate accuracy, while the amount of compression obtained with IGTREE still leads to adequate accuracy. The latter applies also to the G/S task: the compression obtained by IGTREE is 90.4%. Thus, the present results suggest that compression to about 90% of the memory needed by IB1 and IB1-IG is possible without causing generalisation accuracy becoming inadequate.

system	memory (Kb) occupied by algorithm					
	DC	BP	IGTREE	C4.5	IB1	IB1-IG
M (isol.)	4751	59	379	438	4751	4751
A (isol.)	4751	59	305	374	4751	4751
G (isol.)	4751	71	547	7078	4751	4751
Y (isol.)	4751	88	179	120	4751	4751
S (isol.)	4751	88	326	370	4751	4751
M-A-G-Y-S (adap.)	–	461	2691	–	–	–
M-Y-S-A-G (adap.)	–	379	2957	–	–	–
M-G-S (adap.)	–	222	1556	–	–	–
M-S-G (adap.)	–	222	1868	–	–	–
GS	4751	105	759	–	4751	4751
G/S	9502	130	913	–	9502	9502
ART/S	–	–	2984	–	–	–
RND-GS	–	–	1240	–	4751	4751
TYP-GS	–	–	780	–	4751	4751
OCC-GS	–	–	765	–	4751	4751

Table 7.5: Summary of the average memory requirements (in Kb) needed for learning all isolated word-pronunciation subtasks, and all word-pronunciation systems, measured with DC, BP, IGTREE, C4.5, IB1, and IB1-IG. When an algorithm has not been applied to a system, the corresponding cell contains '–'.

Processing speed

While memory usage provides insight into the amount of effort put into compressing the full data base, ranging from none with IB1 and IB1-IG to considerable in BP, it does not provide indications what effort is spent on classification of test instances (i.e., the classification effort discussed in Section 2.1). When measuring the time used by both the training phases and the test phases of all algorithms, indications for both types of effort become available.

We provide an illustration by visualising the time used by DC, BP, IGTREE, IB1, and IB1-IG for (i) learning the GS task, and (ii) classifying the GS test instances, in Figure 7.1. The GS task is taken as example task since it is the most relevant task in view of the problem statement. The figure shows that the effort spent by the algorithms on learning and classification, respectively, indeed correspond, when visualised as coordinate points using logarithmic scales, with the estimated positions of the algorithms in the two-dimensional space spanned up by the learning-effort dimension and the classification-effort dimension as displayed in Figure 2.1 (p. 24). The figure furthermore illustrates the large difference between the summed times needed by IGTREE for learning and classification, viz. about 3020 seconds, as compared to IB1 and IB1-IG, viz. about 15,000 and 16,000 seconds, respectively. In total, IGTREE uses about 20% of the time needed by IB1 and IB1-IG. Finally, the results with BP show that this connectionist-learning algorithm is exceptionally slow on the GS task compared to the other algorithms (the training phase takes about 400,000 seconds). The lengths of learning and classification times were averaged over measurements performed on a Pentium 75Mhz running Linux.

Figure 7.1 bears a resemblance with Figure 2.1: the coordinates of the algorithms in Figure 7.1 are approximately in the same location as the circles circumscribing the same algorithms in Figure 2.1. The latter figure was used to illustrate that each of the three groups of algorithms is located near the end of the learning-effort dimension or the classification-effort dimension. Figure 7.1 can be taken to support the positions of the algorithms as depicted in Figure 2.1 since the points of Figure 7.1 fall within the corresponding ellipses of Figure 2.1.

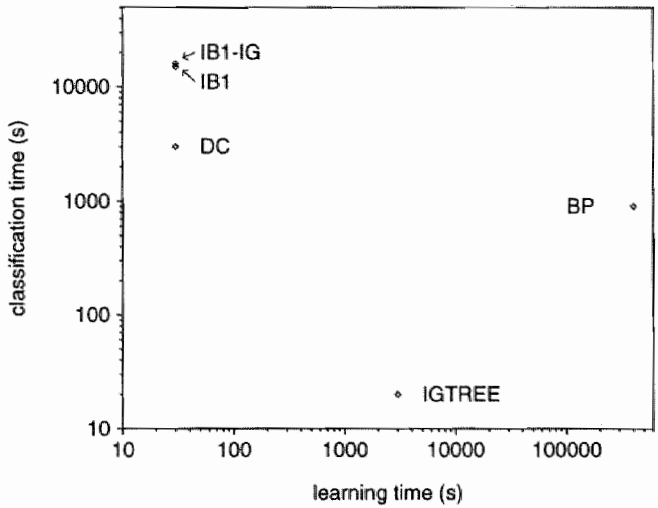


Figure 7.1: Visualisation of the time used by DC, BP, IGTREE, IB1, and IB1-IG on learning the GS task (x-axis), and classifying the GS test instances (y-axis).

7.2 Undiscovered sections in word-pronunciation-system space

Having established which algorithm performs best on which task investigated, it is still impossible to derive a general statement about what would be the optimal decomposition and architecture for a word-pronunciation system. There might well be word-pronunciation systems which perform better than, e.g., IB1-IG trained on the G/S task. In this study we have not focused (i) on optimising systems on speed, or memory requirements, (ii) on constructing modular systems of which the modules are trained with different algorithms, and (iii) on constructing modular systems combining sequential and parallel processing. Although we have argued our choices, it is clear that our study covered a small section of 'word-pronunciation-system' space.

In this section we test three word-pronunciation systems that combine some of the findings presented earlier in this thesis. First, we present G-S, which performs grapheme-phoneme conversion and stress assignment in sequence, and is learned by IB1-IG. Second, we present GS-COMBINER, which combines the outputs of IB1, IB1-IG, and IGTREE on the GS as input to a *combiner* module, trained by IGTREE. Third, we present GS-ARBITER, which combines

the outputs of IB1, IB1-IG, and IGTREE, and the input of the original GS instances as input to an *arbiter* module, trained by IGTREE.

G-S

To demonstrate (and issue a warning) that our coverage of the word-pronunciation-system space has not lead to finding the optimal system, we describe the construction of one particular word-pronunciation system not tested earlier: G-S, which we train with IB1-IG under the adaptive variant. Figure 7.2 displays the G-S architecture. This task combines three earlier findings:

1. There is a small but significant utility effect in incorporating the output of the grapheme-phoneme subtask as input to the stress-assignment subtask, when trained with IGTREE under the adaptive variant (cf. Section modular-summary).

2. Learning a word-pronunciation subtask using the output of a module having performed another subtask appears to be performed best under the adaptive variant (cf. Section 4.5).

3. IB1-IG performs consistently better than IGTREE on all (sub)tasks investigated.

Learning the G-S task with IB1-IG involves constructing a two-modular system of which each module occupies 4,751 Mb. The total memory requirements of 9,502 Mb for the instance bases, combined with the relatively slow classification process of IB1-IG, makes the G-S system not optimal in terms of memory requirements or speed. The three reasons mentioned for testing the G-S system are all related to generalisation accuracy. G-S therefore represents an example exploration in accuracy optimalisation. The reasons mentioned are empirical reasons, but there is also linguistic expert knowledge involved in G-S: expert knowledge is reflected in (i) the decomposition of the two subtasks, and (ii) the accuracy of grapheme-phoneme conversion before stress assignment, as argued for by Allen *et al.* (1987).

Trained under the adaptive variant, i.e., on the collected test output of the grapheme-phoneme-conversion module, IB1-IG obtains a generalisation accuracy of 6.03% incorrectly processed test instances, 3.14% incorrectly classified phonemes, 3.05% incorrectly classified stress assignments, and 33.79% incorrectly processed test words. Table 7.6 lists these accuracy results as well as those obtained on the GS task and the G/S task.

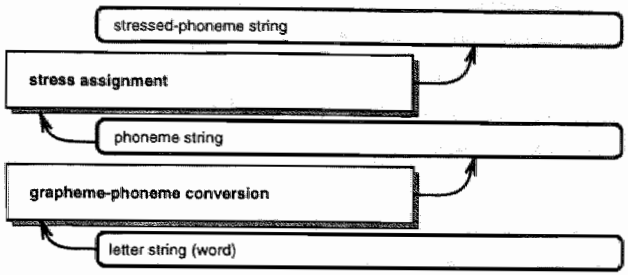


Figure 7.2: Visualisation of G-S, a word pronunciation system containing two modules. Curved-edged boxes indicate input–output representations; sharp-edged boxes denote the two modules. The input–output mappings performed by the modules are depicted by the arrows.

The errors produced on stress assignments in the G-S task are considerably lower than those obtained in the GS and G/S tasks. In the latter two tasks, stress assignment was performed on the basis of letter windows. In G-S, the input consists of (partly erroneous) phonemes. The partly-erroneous input accounts for a worse generalisation accuracy compared to the stress assignment subtask performed in isolation (2.50%, cf. section 3.5), but a better accuracy on stress assignments based on partly erroneous phonemes and morpheme boundaries in the M-G-S system obtained with IGTREE (4.10%). Altogether, the system is able to perform significantly better in terms of the percentage incorrectly processed test instances than IB1-IG on the GS task ($t(19) = 14.12, p < 0.001$), and better than IB1-IG on the G/S task ($t(19) = 10.87, p < 0.001$). These results demonstrate that improvement in generalisation accuracy is possible by combining empirical findings on test material established earlier. However, combining empirical findings obtained on test material in performing new experiments on the same test material violates the constraints on using test material as described in Subsection 2.4.3 (cf. Salzberg, 1995). Therefore, the results obtained with the G-S system are biased and cannot be compared, strictly speaking, with the results presented earlier.

GS-COMBINER and GS-ARBITER

In Chapter 6 we introduced the three gating systems RND-GS, TYP-GS, and OCC-GS. Performing classification tasks in gating systems bears resemblance to

task	% incorrect test instances	phonemes	stress	words
GS	6.82	3.39	3.77	38.88
G/S	6.68	3.14	3.79	36.71
G-S	6.03	3.14	3.05	33.79

Table 7.6: Generalisation accuracies of IB1-IG on the G-S task (bottom line) as compared to those obtained on the GS and G/S tasks (top lines). Results are listed on the percentage incorrectly classified test instances, phonemes, stress assignments, and test words.

recent developments in machine-learning and connectionist-learning research in which multiple classification algorithms are trained on subsets of the full data set. For example, in *bagging*, *boosting*, and *arcing* algorithms (Breiman, 1996a; Freund and Shapire, 1996; Breiman, 1996b) the full data set is resampled during a number of iterations, while learning algorithms are trained on each sample subset (which may be much smaller than the full data set) and the total system's output is generated by combining the outputs of the individual learning algorithms. Using combined outputs of classifiers as input to the same classification task can be viewed as a classification task in itself: systems in which this idea is implemented are generally referred to as *voting* systems (Chan and Stolfo, 1995; Chan and Stolfo, 1997)). Analogous developments can be seen in the research into *ensembles* of neural networks (for an overview, cf. Sharkey, 1997).

In our gating approach the output is not combined; rather, each instance is processed by only one of the modules. Breiman (1996a) and Kai and Boon (1997) express the hypothesis that the combination of the outputs of partially-trained modules will lead to better performance than the non-decomposed system only when the classifications of the different modules are in some way essentially different (e.g., showing a high variance, or showing little overlap in misclassified test instances, Chan and Stolfo, 1997).

We investigate two word-pronunciation systems in which the outputs of IGTREE, IB1, and IB1-IG, trained on the GS task, are combined and used as input to a voting module trained to determine the eventual GS classifications on the basis of the (partly erroneous) classifier outputs. The first system, GS-COMBINER, applies a *combiner* approach to voting (Chan and Stolfo, 1995), meaning in our case that a combiner module is placed in sequence after three

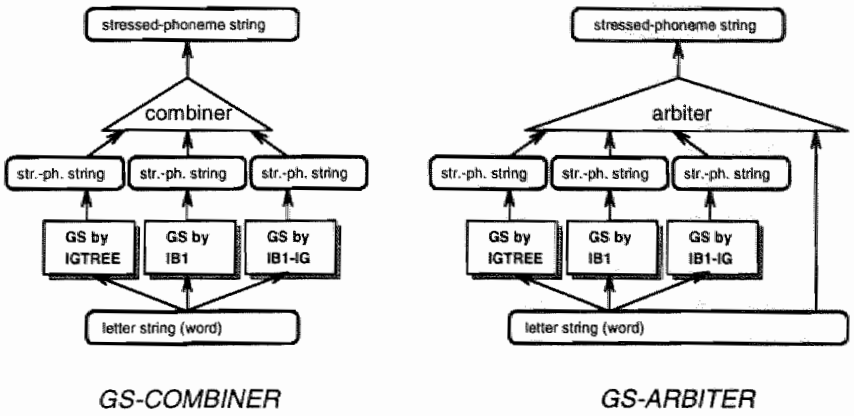


Figure 7.3: Visualisation of GS-COMBINER (left) and GS-ARBITER (right), two word pronunciation voting systems. Curved-edged boxes indicate input–output representations; sharp-edged boxes denote the modules trained by specified algorithms; triangular modules denote the voting modules. Input–output mappings performed by the modules are depicted by the arrows.

modules trained by IGTREE, IB1, and IB1-IG, respectively, taking as input only the (partly erroneous) classification outputs of the three modules. The second system, GS-ARBITER, applies an *arbiter* approach to voting (Chan and Stolfo, 1995). GS-ARBITER introduces an arbiter module which, apart from taking the outputs of the three algorithms as input, also takes the original letter-window instances as input. While the combiner module of GS-COMBINER merely combines outputs without knowledge of the actual letter-window instances, the arbiter module of GS-ARBITER decides which is the best choice for classification given all information. Figure 7.3 displays both systems schematically.

The combiner module of GS-COMBINER and the arbiter module of GS-ARBITER are trained with IGTREE, the algorithm combining small induced models (decision trees) with reasonably fast learning and fast classification. For GS-COMBINER, we constructed a data base of 675,745 instances consisting of three features: the respective classifications of IGTREE, IB1, and IB1-IG on all test instances of the GS task. Analogous to the data sets constructed for the modular systems trained under the adaptive variant (cf. Chapter 4), the instance base is a concatenation of the algorithms' output on test material.

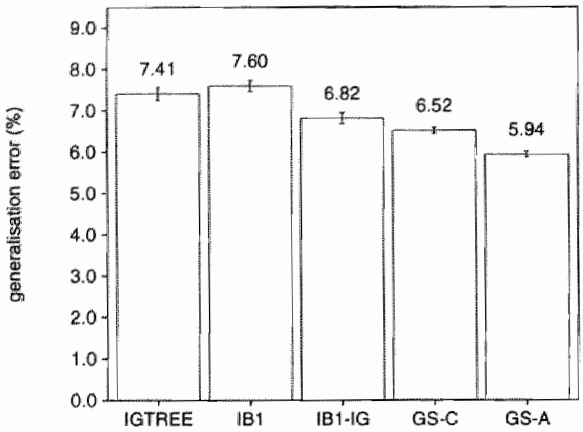


Figure 7.4: Generalisation errors in terms of the percentage incorrectly classified phonemes with stress markers, of the voting systems GS-COMBINER (GS-C) and GS-ARBITER (GS-A), both trained with IGTREE. For comparison, the left part shows the classification errors of IGTREE, IB1, and IB1-IG on the GS task; these classifiers serve as input to both voting systems.

The combiner module is trained in a 10-fold CV setup on this data. The information-gain values of the three features in the data reflects the relative performance differences between the three algorithms on the GS task. The more accuracte the algorithm, the higher the information-gain value of the algorithm: for IGTREE (7.41% generalisation error), it is 4.74; for IB1 (7.60%) it is 4.68; for IB1-IG (6.82%) it is 4.78. The arbiter module of GS-ARBITER was trained in the same setup as the combiner module of GS-COMBINER, with the exception of the instances in the data: apart from the three algorithms' classifications, the instances also include all letter-window information. Figure 7.4 displays the generalisation errors produced by GS-COMBINER and GS-ARBITER.

The results displayed in Figure 7.4 show improvements in generalisation accuracy of both GS-COMBINER and GS-ARBITER over the performance of IB1-IG on the GS task. Both differences are significant; for GS-COMBINER, $t(19) = 6.22, p < 0.001$; for GS-ARBITER, $t(19) = 18.85, p < 0.001$. The generalisation accuracy of GS-ARBITER is also slightly but significantly better than that of IB1-IG on the G-S task ($t(10) = 2.43, p < 0.05$). In terms of incorrectly classified phonemes, GS-ARBITER produces only 2.88% errors. In terms of incorrectly

classified words, however, GS-ARBITER produces more errors (35.80%) than G-S (33.79%).

The results obtained with GS-COMBINER and GS-ARBITER suggest that the outputs of the three classifiers on the GS task differ to such a degree that the combiner and arbiter modules, trained with IGTREE, are able to repair successfully errors made by either of the three algorithms. To analyse this capability of the two voting modules, we counted the numbers of repaired classification errors for different combinations of classification errors, on the performance of both modules on the first partitioning of their 10-fold CV experiments. The results of this analysis, displayed in Table 7.7, show that both voting systems are able to resolve a considerable amount of disagreements between the classifications of IGTREE, IB1, and IB1-IG; for example, the major part of the disagreements in which one of the three classifiers produces an incorrect classification is resolved by both voting systems. An important difference between GS-ARBITER and GS-COMBINER in that the former is able to repair a considerable amount of cases in which all three algorithms produce an incorrect classification. Apparently, GS-ARBITER is able to successfully detect that all three algorithms are incorrect by being able to inspect the letter-window instances; the top row of Table 7.7 shows that this ability of GS-ARBITER also accounts for a small amount of errors on instances for which all three classifications already produced correct classifications.

The results obtained with the two voting systems GS-COMBINER and GS-ARBITER indicate that voting can be profitable for learning word-pronunciation. The three algorithms involved in the two voting systems appear to classify sufficiently differently. The arbiter approach is able to generalise from the algorithms' classifications and the original letter-window instances in such a way that generalisation error is decreased markedly.

7.3 Data characteristics and the suitability of lazy learning

The comparison of generalisation accuracies (Section 7.1) witnesses a consistent superiority of instance-based (lazy) learning. We have concluded that lazy learning is well suited for learning to pronounce written words. In this section we analyse the characteristics of the word-pronunciation task allowing lazy learning to be successful. Previous research on k-nearest neighbour (k-NN) classification, the basis of IB1 and IB1-IG, has addressed a number of limitations and disadvantages of the approach (Breiman *et al.*, 1984; Aha *et al.*,

algorithm correct			# disagreements	# remaining errors	
IGTREE	IB1	IB1-IG		GS-COMBINER	GS-COMBINER
Y	Y	Y	0	1	373
Y	Y	N	995	100	160
Y	N	Y	882	23	111
N	Y	Y	345	41	66
Y	N	N	232	129	108
N	Y	N	354	308	188
N	N	Y	370	267	170
N	N	N	3441	3428	2743
		total	6619	4297	3919

Table 7.7: Absolute numbers of instances leading to eight possible combinations of correct (Y) / incorrect (N) classifications of IGTREE, IB1, and IB1-IG on the GS task, measured on the first partitioning of the data base used for training GS-COMBINER and GS-ARBITER, and the numbers of remaining errors on these eight types of (dis)agreements by both voting systems.

1991; Aha, 1992). Breiman *et al.*, 1984 mention six disadvantages of k-NN classifiers and derived algorithms such as IB1 (the disadvantages are listed in italics, followed by a comment on the applicability of the disadvantage to our study in regular font):

1. k-NN *classifiers are computationally expensive since they save all training instances in memory.* This applies clearly to our study, although our computational resources were sufficient for performing experiments on systems with one or two modules.

2. *They are intolerant of feature noise.* Interpreting this as intolerance of noisy instances, as Aha *et al.* (1991) do, this applies to our data (cf. Section 1.3). However, the apparent success of lazy learning suggests that this is either no disadvantage here, or it is the major source of the otherwise small amount of errors produced with lazy learning applied to our data.

3. *They are intolerant of irrelevant features.* This applies to IB1, but not to IB1-IG insofar as information-gain weighting can detect irrelevant features in our data.

4. *They are sensitive to the choice of the algorithm's similarity function.* This is true, but information-gain weighting is a relatively unbiased (and, for our data, successful) data-oriented method for estimating weights of features. Many other data-oriented similarity functions exist (Wettschereck *et al.*, 1997).

5. *There is no natural way to work with nominal-valued features or missing features.* This applies to our nominal-valued data, but we show that our distance function, combined with information-gain weighting, can lead to adequate similarity matching and consequent adequate accuracy.

6. *They provide little usable information regarding the structure of the data.* Indeed, no direct insight can be gained from performance results of IB1 and IB1 into inherent data characteristics.

In the following we propose and perform a procedure for discovering the characteristics of our data that make it especially suitable for lazy learning, and enable lazy learning to soften or avoid the disadvantages noted by Breiman *et al.* (1984). First, we describe such a procedure proposed by Aha (1992). Then, we propose a procedure to be performed on our data, inspired by that of Aha (1992).

Aha (1992) proposes a method for discovering which characteristics of the data relate to the applicability of certain learning algorithms to this data. The method is comprised of three phases:

1. In the first phase, an artificial data set is constructed (Benedict, 1990) mimicking the investigated (real-world) data set in terms of learnability.

2. In the second phase, algorithms are trained and tested on the artificial data set of which the characteristics are gradually shifted.

3. In the third phase, a factorial analysis in the form of rule induction (Clark and Niblett, 1989) is applied to the experimental outcomes of the second phase, yielding rules such as *IF (the number of training instances is larger than x) AND (the number of relevant features is larger than y) THEN algorithm A will produce significantly better generalisation accuracy than algorithm B* (Aha, 1992).

This method assumes that when one has found an artificial data base mimicking the learnability of the original data base, that the known characteristics

of that data base largely overlap with those of the original data base. However, as Aha (1992) notes, when assembling an artificial data base on the basis of known characteristics, one may miss out on an undiscovered yet essential characteristic of the original data set.

Inspired by Aha's (1992) method we propose a data-characterisation procedure applied to the GS data. The procedure aims to discover which primary characteristic of the GS data allows IB1 and IB1-IG to perform adequately. The procedure is limited in that it does not operate on the basis of an artificial data set of which the characteristics are well known. Rather, it operates on data sets in which one aspect of the original GS instance base is distorted. The reasoning behind this limited procedure is that apart from the surface attributes of the instance base (e.g., number of features, values per feature, instances, classes), we have thus far not gathered any indications on the number of clusters (i.e., groups of nearest-neighbour instances belonging to the same class), the number of disjunct clusters per class (i.e., the numbers of separate clusters per class), and the numbers of prototypes per class (Aha, 1992). Prototypes can be defined as centroids, i.e., vectors at the centre of clusters in instance space (viz., the space spanned up by all possible feature-value vectors representing the task) (Benedict, 1990). We have no indications on the number of prototypes in our data, and can only refer for comparison to studies on other data indicating that "IB1 performs well for highly disjunctive target concepts" (Aha, 1992, p. 6). The more prototypes an instance base contains, the more disjunct class clusters exist: the general expectation is that IB1 will perform generally better than eager-learning algorithms on such data since it retains all information concerning disjuncts, no matter how small, while decision trees (notably those implementing pruning) tend to overgeneralise and miss out on disambiguating small disjuncts (Holte *et al.*, 1989; Ali, 1996; Danyluk and Provost, 1993; Provost and Aronis, 1996; Aha, 1992, provides some careful modifications of this expectation). Our working assumption here is, by abduction, that our data contains many small disjuncts, since IB1 and IB1-IG perform better than IGTREE, C4.5, and BP. Apart from investigating the characteristic that favours lazy learning over eager learning, we also aim to collect indications on the characteristic that favours IB1-IG over IB1.

The procedure is structured as follows:

1. We construct two data sets mimicking the original data base representing the GS task, in which one aspect of the data is altered:

   (a) In the first data set, called GS-PERM, we permute randomly for each word all letters along with their corresponding PSs; i.e, for each

word, all letter-phoneme correspondences are shuffled. GS-PERM distorts the context around focus letters, but leaves the correspondences between the focus letters and phonemes intact.

(b) The second data set, called GS-RAND, randomises for each word all letters. While the phonemic transcription with stress markers is maintained, all letters of the word are randomly picked from the letter alphabet. This distorts all letter-phoneme correspondences, and makes the relation between spelling and pronunciation fully arbitrary (not unlike ideographic writing systems).

2. IGTREE is applied to the GS, GS-PERM, and GS-RAND data sets, rendering for each data set a decision tree. For each tree, the following statistics are computed:

   (a) the numbers of leaves at each level of the tree (i.e., the numbers of clusters discerned by IGTREE at different levels) and

   (b) for each level of the tree, the average numbers of instances represented by each leaf (i.e., the sizes of clusters discerned by IGTREE at different levels).

3. Leave-one-out experiments (Weiss and Kulikowski, 1991) are performed with IB1 and IB1-IG on the GS, GS-PERM, and GS-RAND data sets. In each experiment, for each instance the number of nearest neighbours of the same class is determined, i.e., the number of *friendly* nearest neighbours of an instance. Because IB1 and IB1-IG employ different distance functions (cf. Subsection 2.1.3), their average numbers of friendly neighbours may differ and may reveal why IB1-IG performs better than IB1 does.

Searching for small disjuncts

Measures 2(a) and 2(b) attempt to provide broad indications of cluster sizes by counting disambiguated instances at different levels in trees constructed by IGTREE. During the construction of a tree IGTREE is assumed to be disambiguating as many instances as possible, as early in the tree as possible. The strategy of IGTREE is to detect clusters of instances and representing them by paths ending at leaf nodes. When IGTREE can, for example, construct a path ending in a leaf node at level three, representing the disambiguated classification of 100 instances, it has discovered three feature-value tests bounding a

level	average # instances per leaf								
	GS			GS-PERM			GS-RAND		
1	5	6.91	± 2.73	5	6.91	± 2.73	0	–	–
2	172	15.56	± 3.72	145	3.20	± 1.16	0	–	–
3	3413	16.02	± 2.87	4084	3.96	± 1.22	413	1.35	± 0.10
4	18842	8.08	± 0.98	75727	1.42	± 0.13	436860	1.01	± 0.01
5	32017	5.46	± 0.74	250061	1.11	± 0.06	126803	1.01	± 0.00
6	22565	3.85	± 0.18	123941	1.10	± 0.02	65133	1.01	± 0.01
7	24208	6.36	± 0.95	94068	1.33	± 0.25	46417	1.02	± 0.02

Table 7.8: Numbers of leaves and average numbers of instances represented by these leaves (with standard deviations), for each of the seven levels in trees constructed by IGTREE on the GS, GS-PERM, and GS-RAND instance bases.

cluster of instances of size 100 of the same class. The assumption underlying the clustering of instances at leaf nodes in IGTREE is that information gain, computed over the full instance base, provides an adequate approximation of the ordering of features to be investigated for maximising the numbers of disambiguated instances as high as possible in the tree[4]. This assumption constitutes a bias in any conclusion drawn from measures 2(a) and 2(b).

Table 7.8 lists the numbers of leaves and average numbers of instances represented by those leaves, per level, produced by IGTREE on the full GS, GS-PERM, and GS-RAND instance bases. A straightforward difference between the trees generated on the full GS instance base on the one hand, and the GS-PERM and GS-RAND instance bases on the other hand is found in the total numbers of leaves: 101,222 (GS), 547,995 (GS-PERM), and 675,626 (GS-RAND). GS-PERM and GS-RAND represent distortions of the GS task that cause IGTREE to build very large, cumbersome trees. The average number of instances represented by leaves at all levels is considerably larger for the GS tree than for the GS-PERM and GS-RAND trees. Two attributes of the GS tree appear salient: first, at levels 2 and 3 of the GS tree, IGTREE can form clusters of (on average) 16

[4] C4.5 arguably provides a better approximation than IGTREE since it adjusts the information-gain ordering of features at every non-ending node in order to maximise the numbers of instances that can be disambiguated as high in the tree as possible. Unfortunately, it was not feasible to run C4.5 on the GS task given our maximal computational resources.

instances of the same class. Thus, by deciding on two or three feature-value tests, approximately 3,500 clusters of size 16 (on average) can be identified; this clustering accounts for a considerable amount of compression. Second, at deeper levels in the GS tree cluster size decreases, though on average it remains at three instances or more. For example, on level 5, clusters contain on average approximately five instances (the small standard deviations in Table 7.8 indicate that the average number of instances per cluster per level is quite stable). In contrast, paths in the GS-PERM tree from level 4 onwards, and in the GS-RAND tree altogether, represent only single instances. IGTREE fails to detect clusters in the GS-PERM and GS-RAND data and resorts to assigning unique paths to almost all instances (about 98% of the GS-PERM instances, and virtually 100% of the GS-RAND instances). We conclude from the results discussed here that, under the assumption that IGTREE performs an adequate clustering of the data (which is biased by the specific choice of information-gain feature ordering), the instances in the GS instance base are clustered in small disjuncts of size three to sixteen, on average. It is likely that given enough training instances, one of the minimal two or three instances of a disjunct are stored in memory – each of these single instances can then serve as the perfect match for th instances in its disjunct in the test material.

Daelemans (1995, 1996a, 1996b) argues that within instance bases representing language tasks such as the ones studied here, *pockets of exceptions* can be found; "Exceptions tend to come in 'families' " (Daelemans, 1996a, p. 5). Our analysis with IGTREE suggests that these families typically have three or more members. We join Daelemans in his statement that a learning algorithm applied to a language tasks that can be rephrased as classification tasks should focus on storing all instances because of this specific feature. Therefore, storing all instances is a favourable property of IB1 and IB1-IG; approaches which stop storing information below a certain utility threshold, ignoring small disjuncts, such as decision-tree learning with pruning (Quinlan, 1993; Holte *et al.*, 1989) lack this property. Nevertheless, there is still need for an argument why IGTREE and C4.5 without pruning perform worse than IB1 and IB1-IG, because both decision-tree algorithms expand their trees to represent any small disjuncts. We assert that this is because both algorithms assume a feature ordering that is too rigid. A mismatch on a feature-value test early on in tree traversal blocks any other feature-value tests at deeper levels that may match the test instance and lead the way to a small disjunct with the possibly correct classification. Indeed, the information-gain values of the outer context letters in the window computed for the GS task, display

little differences (cf. Appendix D) – a bad motivation for ordered testing of feature values. To avoid such mismatches, at least two algorithmic solutions other than the one offered by IB1-IG appear salient; (i) a hybrid algorithm mixing IGTREE and IB1-IG, in which IGTREE is invoked up to the point where tree construction becomes badly motivated, after which IB1-IG is performed on the remaining informationally-insignificant features[5], and (ii) lazy decision trees (Friedman, Kohavi, and Yun, 1996), in which a test instance is classified by searching an instance base and constructing an imaginary decision tree path specifically tuned to this instance, attempting to avoid matches on features not relevant to the instance.

As a aside, we remark that the slight increase in leaves and instances per leaf at level 7 of the GS and GS-PERM trees displayed in Table 7.8, reflects that at that level, certain ambiguities are not resolved. Level 7 contains an extra number of instances not yet disambiguated that need further feature-value tests or are inherently ambiguous. This can partly be resolved by expanding the window (cf. Section 7.5).

From the perspective of the task it is useful to gather information on the content on the clusters to assess if they reflect an inherent characteristic of the data. Dealing with approximately 95,000 clusters and 675,000 instances distributed over these clusters, averaging analyses may provide adequate indications answering this question, although such analyses will ignore possibly many interesting subtle facts. We performed two averaging analyses, focusing on the correlation between clusters of instances represented by leaves of the GS-tree, and (i) morphological-segmentation classification of these instances, and (ii) syllabification classification of these instances. An instance is marked by a morphological segmentation when its classification in the morphological-segmentation subtask (cf. Section 3.1) is '1', e.g., the instance **ooking_**, derived from the sample word **booking** in which the **i** marks the beginning of the inflectional morpheme **ing**. Analogously, an instance is marked by a syllable segmentation when it maps to class '1' in the syllabification subtask with letters as input, such as the instance **booking**, in which the **k** is marks the beginning of the syllable /kɪŋ/. For each cluster of instances we computed the distribution of morphological-segmentation classes and syllabification classes of these instances in the instance bases of the respective subtasks. The bias underlying this analysis is the assumption

[5] An instantiation of this hybrid IGTREE/IB1-IG idea is the TRIBL algorithm described by Daelemans, Van den Bosch, and Zavrel 1997b. In the context of this thesis, TRIBL is future research.

	boundary marker on focus			
	morphological		syllabic	
level	absent ('0')	present('1')	absent ('0')	present ('1')
1	**100.0**	0.0	**100.0**	0.0
2	**76.5**	23.5	**92.7**	7.3
3	**75.8**	24.2	**86.8**	13.2
4	**80.8**	19.2	**75.1**	24.9
5	**76.1**	23.9	**77.6**	22.4
6	59.9	**40.1**	47.8	**52.2**
7	53.0	**47.0**	54.4	**45.6**
instance base	71.0	29.0	68.8	31.2

Table 7.9: Percentage distribution of morpheme and syllable boundaries ('0', absent, or '1', present) occurring at the focus position of instances represented by leaves, for all levels of the tree constructed by IGTREE on the GS instance base. The bottom line lists the overall distributions of the markers in their respective instance bases. Percentage distributions larger than the overall distributions are printed in bold.

that there might be a global correlation between clusters at different levels in the tree, and morphological and syllabic markedness of instances in these clusters. We restrict the notion of markedness to refer to the presence of a non-null morphological-segmentation or syllabic classification of the focus letter of each instance. Morphological-segmentation and syllabic classifications ('0' and '1') are taken from the instances bases for the respective isolated subtasks (cf. Subsections 3.1 and 3.4) The results of the analyses are displayed in Table 7.9. For each level, the percentage morphological and syllabic classifications are computed of instances represented by leaves at that level. For comparison, the bottom line of Table 7.9 lists the overall distribution of classes in the full instance bases of morphological segmentation and syllabification.

Globally, the results in Table 7.9 indicate that at the highest levels in the tree, most instances represented by leaves represent a letter window in which the focus letter is not at the initial position of a morpheme or a syllable. The deeper in the tree, the more instances represent letter windows of which the focus letter is in initial position of a morpheme or syllable. This is witnessed by

an above-average number of instances mapping to class '1' at the bottom two levels. Both morphologically and syllabically-marked instances occur relatively more frequently in instances represented at the bottom two levels in the tree than they do in the full instance base. From these results we can conclude that class ambiguity can be resolved by taking into account less context for within-morpheme and within-syllable instances than for instances marking boundaries. These results indicate the relevance of morphemes and syllables in word pronunciation (Allen *et al.*, 1987; Daelemans, 1987; Daelemans, 1988; Coker *et al.*, 1990; Vroomen and Van den Bosch, to apear). We argue that it is an argument for considering morphological and syllabic information as good descriptions of the structure of words and pronunciations, but not necessarily for considering them as necessary abstraction levels in word pronunciation. Our overall negative findings on including morphological segmentation and syllabification in word pronunciation (Chapter 4), together with our experiments on occurrence-based gating (correlating roughly with morphological structure, Section 6.3) suggest that morphological and syllabic structure can be left implicit in the mapping from spelling to pronunciation. As far as morphological and syllabic structure is necessary for determining word pronunciation, the word-pronunciation data itself will reflect it and learning algorithms are capable of tracing it when necessary (to an adequate degree), as Table 7.9 indicates for the case of IGTREE.

An alternative method of detecting clusters, less biased than the information-gain ordered cluster detection by IGTREE, can be implemented using IB1. We performed leave-one-out experiments (Weiss and Kulikowski, 1991) in which we computed for each instance in the GS, GS-PERM, and GS-RAND data sets a ranking of the 100 nearest neighbours in terms of their distance to the left-out instance. Within this ranked list we count the ranking of the nearest neighbour of a different class than the left-out instance. This rank number minus one is then taken as the cluster size surrounding the left-out instance. If, for example, a left-out instance is surrounded by three instances of the same class at distance 1 (i.e., one mismatching feature-value), followed by a fourth nearest-neighbour instance of a different class at distance 2, the left-out instance is said to be in a cluster of size three. The results of the three leave-one-out experiments are displayed graphically in Figure 7.5.

The x-axis of Figure 7.5 denotes the numbers of friendly neighbours found surrounding instances; the y-axis denotes (in logarithmic scale) the occurrences of friendly-neighbour clusters of particular sizes. There are distinct differences between the three scatter plots representing the three data sets.

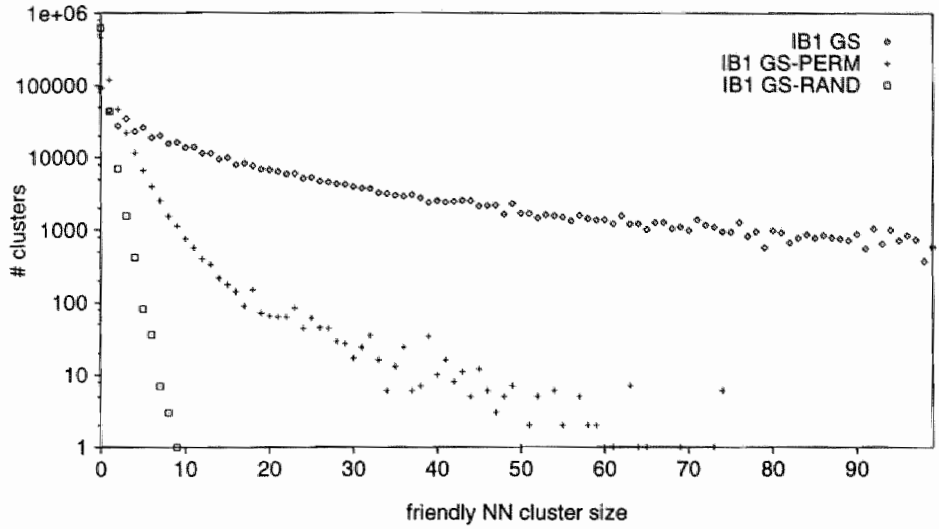


Figure 7.5: Scatter plots of numbers of friendly-neighbour clusters of sizes 0 to 99, as found by IB1 on the GS, GS-PERM, and GS-RAND data sets.

With the GS-RAND data set most friendly-neighbour clusters are of size 0 (i.e., the nearest neighbour is associated with another class), some are of size 1 to 5, and accidentally clusters of 6, 7, or 8 friendly neighbours are found. This is to be expected in a data set of which the feature values are randomised: there are no similar feature-value vectors associated with the same class, unless in accidental cases. With the GS-PERM data set, the occurrence of clusters is not very different from those in the GS-RAND data set. Somewhat less random feature-value vectors occur in the permuted GS-PERM data set; there are accidental friendly-neighbour clusters with more than ten instances. Both scatter plots contrast strongly with the scatter plot computed for the GS data set. Within the GS data, IB1 finds thousands of friendly-neighbour clusters of sizes 1 to 60, and still hundreds of clusters containing more than 60 friendly neighbours. Combining these results obtained with IB1 with those obtained with IGTREE, we can conclude that the GS data contains many thousands of small disjunct clusters containing about three to about a hundred instances each.

As an aside, we note that we did not perform empirical tests with IB1 and IB1-IG using k-NN with $k > 1$. Therefore we cannot claim that 1-NN is the most optimal setting for our experiments. The results discussed in this section

data set	average # friendly neighbours IB1	IB1-IG
GS	15.01	25.58
GS-PERM	0.50	3.41
GS-RAND	0.11	0.11

Table 7.10: Average numbers of friendly (identically-classified) nearest neighbours of GS instances, measured with IB1 and IB1-IG.

suggest that the average 'k' actually surrounding an instance is larger than 1, although many instances have only one or no friendly neighbour. The latter suggests that a considerable amount of ambiguity is found in instances that are highly alike; matching with k-NN with $k > 1$ may fail to detect those cases in which an instance has one best-matching friendly neighbour, and many near-best-matching instances of a different class. Empirical tests are needed to estimate the effects of larger k.

Information gain: adapting the distance function to the data

Having collected broad indications for the degree of disjunctive clusteredness of the data, which generally favours lazy learning over greedy approaches (Holte *et al.*, 1989), we now turn to locating a relevant characteristic of our data favouring IB1-IG over IB1. An argument for including IB1-IG in our study was that IB1-IG is equipped with a weighting function sensitive to relevance differences of individual features (cf. Subsection 2.1.3). To compare IB1 and IB1-IG we performed the same leave-one-out experiment with IB1-IG as described above to compute the numbers and sizes of friendly-neighbour clusters discerned by IB1. Table 7.10 displays the average size of friendly-neighbour clusters found by IB1 and IB1-IG in the three data sets (averaged over all instances per data set).

Table 7.10 provides three relevant indications. First, comparing IB1 and IB1-IG on the GS task, it can be seen that larger clusters of friendly neighbours are found with IB1-IG than with IB1. Thus, by weighting the distance function (cf. Eq. 2.1), IB1-IG rescales the instance space in such a way that instances of the same class become surrounded with more instances of the same class as compared to when the similarity function of IB1 is used. The advantage of the information-gain-weighted distance function supports the assumption un-

derlying IB1-IG (cf. Subsection 2.1.3), that information gain is a good method for estimating feature relevance. Second, the results in Table 7.10 on the GS-PERM data set indicate that with IB1 the average friendly-neighbour cluster size is smaller than 1, i.e., most nearest neighbours are of a different class. With IB1-IG, however, the average friendly-neighbour cluster size is 3.41: on average each instance is surrounded by over 3 instances of the same class. It appears that perturbing the context around the focus letter causes the flat-weighted distance function of IB1 to lose the ability to discern (on average) between instances of the same class and instances of different classes. In contrast, because the information-gain-weighted distance function of IB1-IG still recognises that the focus letter is highly relevant to classification, IB1-IG still rescales the instance space making instances of the same class (on average) more similar to each other than instances of different classes. Third, when the correspondence between focus letters and phonemes is lost and information gain cannot detect any relevance differences between features, which is the case in the GS-RAND data set, both IB1 and IB1-IG fail to measure differences in average distance between instances of the same class and between instances of different classes.

In sum, the feature values of word-pronunciation instances, computed on the full data set, display outspoken relevance differences (the middle letter of a window being by far the most relevant for classification); by adapting the distance function in IB1 to this intrinsic global characteristic of the data, IB1-IG employs an empirically good estimate of distance between instances. Indirectly, together with the adequate accuracy of lazy learning on word pronunciation, this shows that the features used for representing the input for the word-pronunciation task, i.e., fixed-sized windows of letters, are adequate for lazy learning. There appears to be no direct need to construct meta-features to group together clusters in instance space (Aha, 1991). Word-pronunciation can be learned adequately by merely counting matching letters within a bound, local context (cf. Section 7.5 for a discussion on extending the context).

7.4 Related research in machine learning of morpho-phonology

The present study cannot be viewed but in relation with the growing interest in applying machine-learning algorithms to natural language processing (NLP) tasks (e.g., Powers and Daelemans, 1991; Oflazer and Somers,

1996; Daelemans *et al.*, 1997c)[6]. Recent applications of ML algorithms to morpho-phonological tasks do not show an exclusive use of instance-based (lazy) learning algorithms such as IB1 or IB1-IG. Rather, our study and related work (Stanfill and Waltz, 1986; Lehnert, 1987; Wolpert, 1990; Weijters, 1991; Van den Bosch and Daelemans, 1992; Van den Bosch and Daelemans, 1993; Daelemans and Van den Bosch, 1992a; Van den Bosch *et al.*, 1995; Daelemans and Van den Bosch, 1997; Wolters, 1997; Wolters and Van den Bosch, 1997) are the only known reports on applying instance-based learning algorithms to morpho-phonological tasks. However, there is a considerable body of work on applying eager learning algorithms (e.g., decision-tree learning, connectionist learning) to morpho-phonological tasks.

The classical NETTALK paper by Sejnowski and Rosenberg (1987) can be seen as a primary source of inspiration for the present study; it has been so for a considerable amount of related work. Although it has been criticised for being vague and presumptuous and for presenting generalisation accuracies that can be improved easily with other learning methods (Wolpert, 1990; Weijters, 1991; Yvon, 1996), it was the first paper to investigate grapheme-phoneme conversion as an interesting application for general-purpose learning algorithms. The NETTALK data has reappeared in many studies on machine-learning algorithms (Stanfill and Waltz, 1986; Lehnert, 1987; Dietterich *et al.*, 1995; Weijters, 1991) and has reached the status of a benchmark problem (Murphy and Aha, 1995). A recent study by Yvon (1996) still bases its conclusions partly on the learnability of grapheme-phoneme conversion by a machine-learning algorithm on results obtained with the NETTALK data. Our study is based on a corpus of about four times the size of NETTALK, and our investigation of learning the word-pronunciation task based on half our corpus with the RND-OCC task (Section 6.1) showed a marked decrease in generalisation accuracy. The NETTALK data appears too small to be used in an experiment aiming at obtaining adequate generalisation accuracy on grapheme-phoneme conversion. Empirical tests would be needed to investigate whether different corpora of English word pronunciations yield comparable results when sizes are comparable to the CELEX or NETTALK corpora.

Examples of similar experiments performed with connectionist learning algorithms on morpho-phonological data, aimed at estimating and optimising generalisation accuracy or demonstrating the efficacy of connectionist

[6] For more details on machine learning of natural language, the reader is referred to the SIGNLL web page of collected links at URL http://www.cs.unimaas.nl/signll/signll-www.html.

learning, are the aforementioned Dietterich *et al.* (1995) and Wolters (1996) (both on grapheme-phoneme conversion), Brunak and Lautrup (1990) (on English hyphenation), and Ling (1994) (on past-tense learning of English verbs, a task closely related to phonology). An example of modelling explicit linguistic expert knowledge is provided by Gupta and Touretzky (1992), who investigate the learnability of stress in nineteen languages, comparing the learnability of an explicit encoding of metrical phonological theory (Dresher and Kaye, 1990) and the learnability of an unbiased task definition. Gupta and Touretzky (1992) conclude that both approaches behave similarly, and assert that neither can be regarded as providing insight into the intrinsic structure of stress in the tested languages. Psycholinguistically-motivated models of reading aloud are most commonly trained and tested on relatively small corpora (of NETTALK size or less) (Coltheart *et al.*, 1993; Norris, 1993; Bullinaria, 1993; Plaut *et al.*, 1996). Their goal is generally to model human behaviour, which imposes very tight restrictions to the material used (e.g., small data set, monosyllabicity, fixed numbers of letters and phonemes, no complex stress, no morphological structure).

Although the goals of these models are diverse they do share some common assumptions, of which we mention explicitly the windowing method. This method can be found with nearly all afore-mentioned studies, except for the tasks which take whole words as input (Ling, 1994; Gupta and Touretzky, 1992). An interesting counterargument, put forward by Yvon (1996), is that an inductive-learning approach to grapheme-phoneme conversion should be based on associating variable-length chunks of letters to variable-length chunks of phonemes. This chunk-based approach contrasts with the windowing method which supposes fixed-length chunks of letters being associated to single phonemes. The chunk-based approach is shown to be applicable, with adequate accuracy, to several corpora, including corpora of French word pronunciations and, as mentioned above, the NETTALK data (Yvon, 1996). The approach appears to be sensitive in a negative sense, however, to morphological complexity in languages such as English, and appears to be particularly sensitive to sparse data, to which instance-based learning is less sensitive (Wolters and Van den Bosch, 1997). Future experiments on larger amounts of data are needed to investigate the potential advantages or disadvantages of the chunk-based approach over our phoneme-based approach.

Related issues in computational morphology

Mainstream work in computational morphology focuses on developing morphological analysers of which the processing is based on finite-state automata, operating in combination with a lexicon containing all morphemes of the target language (Koskenniemi, 1984; Allen *et al.*, 1987). After providing an overview of the mainstream work in computational morphology, Sproat (1992) notes that (i) the state of research in computational morphology in 1992 calls for future work on models less dominated than the mainstream work by finite-state multiple-level morphotactics (as their complexity is unfavourable, unless parallel machines could be used (Koskenniemi and Church, 1988)). Moreover, he notes that (ii) more work needs to be done in the area of modelling human morphological processing than done in the mainstream work (Sproat, 1992). Finally, he remarks that (iii) morphological analysis in some generation systems (e.g., text-to-speech conversion) should be integrated better with phonology than it is done in mainstream work (Sproat, 1992). Our study addresses Sproat's (1992) concerns (i) and (iii) directly, although partly. We investigate morphological analysis as a straightforward one-pass segmentation task (Section 3.1) and address the incorporation of morphological analysis as a subtask of word pronunciation (Chapter 4).

Related issues in computational phonology

The development of phonological theory has mainly diverged on how to represent phonological rules. Chomsky and Halle's (1968) book *The Sound Pattern of English* (SPE) marked the first decade of generative phonology, a framework in which phonological rules are represented as linearly ordered, context-sensitive rules. However, the complexity of the placement of stress markers calls for non-linear processing, which has led to the development of two theories of non-linear phonological processing: autosegmental phonology or two-level phonology (Goldsmith, 1976; Bird, 1996) and metrical phonology (Liberman and Prince, 1977) (cf. Subsection 2.2.2). Bird (1994) discerns four strands of current work in computational phonology: (i) providing formal frameworks in which phonological theories can be expressed; (ii) implementing computer programmes to be used by phonologists for developing and testing phonological theories; (iii) developing models for simulating human (phonological) behaviour, to simulate the acquisition of phonological knowledge, and to present phonologists with useful generalisations about a certain body of phonological data; and (iv) integrating computational mod-

els of phonology with computational models of syntax and speech. The strand most relevant to our work is the third (Dresher and Kaye, 1990; Gasser and Lee, 1990; Gupta and Touretzky, 1992; Ellison, 1993; Daelemans *et al.*, 1996; Gillis *et al.*, 1995; Daelemans *et al.*, 1994a), which use the actual phonological data as occurring in daily usage or in dictionaries as the basis for acquiring (inducing) models of phonology. The approach taken here is a representative of this strand of work, with the exception that most of the cited work is biased towards incorporating linguistic expert knowledge in the input and/or output of the investigated phonological tasks, while we have focused on avoiding the incorporation of linguistic expert knowledge maximally to test how much expert knowledge may be left implicit in the data.

7.5 Limitations, extensions, and future research

Inductive language learning spans a huge experiment space of learning algorithms and language tasks of which the present study has explored a subspace. We have argued the relevance of searching this particular subspace in Chapters 1 and 2. We mention interesting and relevant extensions of the present approach in the experimental space. We group the overview of extensions on the basis of some general limitations of our approach.

Extending the window

Windowing is limited to seven letters per instance, for all experiments described in this thesis. We assumed this context would be sufficient for attaining adequate generalisation accuracy, and named this the locality assumption. The information-gain values computed for the seven positions in the window indicated that for tasks such as grapheme-phoneme conversion and GS, the outer context letters received rather low information-gain weights (cf. Appendix D). However, a notable amount of error stems directly from the fact that a context of three left characters and three right characters does not suffice to disambiguate between alternatives (consider, for example, the two different phonemes and stress markers associated with the first **o** of **photograph** and **photography**). The errors on test material arising from these ambiguities were measured to range between 1.25% (with morphological segmentation) and 2.09% (with the GS task), accounting for about 27% (with morphological segmentation) to 31% (with GS) of the total number of generalisation errors.

task	% incorrect			
	instances	phonemes	stress	words
GS, 3-1-3	3.86	1.31	2.73	24.82
GS, 4-1-4	2.10	0.58	1.61	14.63
GS, 5-1-5	1.20	0.26	0.98	8.85
GS, 6-1-6	0.71	0.12	0.61	5.48

Table 7.11: Reproduction (upper-bound) accuracy, in terms of percentage incorrectly classified instances, phonemes, stress assignments, and words, of IB1-IG trained and tested on the full data set of the GS task using 3-1-3, 4-1-4, 5-1-5, and 6-1-6 windows, respectively. window, respectively.

Although the pronunciations of homographs such as **read**, **suspect**, and **object** cannot be disambiguated by an approach that investigates isolated words, a major part of the ambiguities caused by the insufficient window width can be solved by extending the window to the maximal width needed for the most exceptional case. Extending the window can be incorporated easily with the different algorithms, but the consequences of extending the window differ per algorithm. The reasonably compact MFNs to which BP is applied become larger with a larger window, and BP can be expected to spend considerably more time learning. For the instance-based algorithms, a window expansion would imply a serious (i.e, exponential) setback in classification speed. For IGTREE and C4.5, however, only the paths that are ambiguous at depth seven of the tree are expanded to disambiguate the mapping, thus, both algorithms can be expected to suffer only marginally from expanding the window.

To provide indications of the effects of extending the window width, we analyse the theoretical upper-bound accuracy on the GS task (computed as described in Section 7.1, with IB1-IG). We performed experiments on the GS task using nine-letter, eleven-letter, and thirteen-letter windows, respectively with four, five, and six letters on either side of the focus letter. The resulting upper-bound generalisation errors measured after training and testing on all instances, phonemes, stress assignments, and words in the data set are listed in Table 7.11. The accuracy results suggest strongly that improvement is indeed possible by extending the window. The results also indicate

that the amount of true ambiguity in the data occurring with homographs is quite small compared to the amount of ambiguity that can be resolved by expanding the window, causing at most 5.48% incorrectly processed words, i.e., approximately one fourth of the upper-bound errors on the GS task using a seven-letter window. Empirical tests (10-fold CV experiments) are needed to determine whether the extension of the window may indeed bring generalisation accuracy further under the 5% lower threshold error on phonemes, and under 20% error on test words.

Word pronunciation in texts

Word pronunciation is, in our definition, the pronunciation of words in isolation. We have not addressed word pronunciation in textual context. Encoding words in textual context can, however, easily be incorporated into the approach (for an example, cf. Weijters and Hoppenbrouwers, 1990), by letting the window run across text. Instead of filling the left and right context of each word with word boundary markers ('_'), the window incorporates both a part of the left-hand-side word, the single space between the words, and a part of the right-hand-side word. Within textual context, pronunciations of words are sometimes blurred by phonemic alternations at the word boundaries, similar to phonological processes occurring at certain morpheme or syllable boundaries (cf. Subsection 2.2.2).

While the approach can be copied effortlessly, the task changes dramatically. We treat our dictionary words as occurring only once, which is a limitation given the fact that there are large differences in the frequencies of word tokens in text. Word frequencies may play an important role in learning word pronunciation: shorter words tend to reoccur more often than longer words (Zipf, 1935); many of the most frequent words in English are pronounced irregularly (Andrews, 1992). Moreover, text generally contains a considerably larger portion of monomorphemic words than a dictionary, as any comparison between a regular piece of text with a list of dictionary entries readily shows.

Applying our approach in a text-to-speech synthesiser aimed at pronouncing text would, of course, require a data base of transcribed text. Empirical tests are needed to determine whether word pronunciation in text is learned with better generalisation accuracy than isolated word pronunciation.

Incorporating previous classifications in the input

In the present approach classification of instances is independent of the classification of other instances, e.g., belonging to the same word. The approach allows for instances to be classified in any order, including the most obvious order, viz. from left to right (the direction of reading). When classification is explicitly and strictly limited to the left-to-right order, the possibility arises to incorporate previous classifications into the input of all instances of a word, except into the first, leftmost instance (Weijters, 1990; Daelemans, 1996, p.c.). In the example of grapheme-phoneme conversion, one could implement the classification of a phoneme as done on the basis of seven letters and three previously classified phonemes. Since the phonotactics of English are quite strict as regards the adjacency of phonemes (cf. Subsection 2.2.2), it can be expected that the previously classified phonemes would be relevant in determining their next neighbour and that they will receive quite high information-gain values.

A potential drawback of the approach is the possibility of cascading errors occurring with test words; i.e., when one test instance is classified incorrectly, its incorrect classification is present in the input of the three consecutive instances, which may lead to continuously cascading errors due to continuously incorrect input. Adamson and Damper (1996) trained a recurrent Jordan neural network (Jordan, 1986) on the grapheme-phoneme task for British English; the network copies its output (i.e., the phoneme) into the input layer. Experiments showed worse generalisation accuracy by the Jordan network than by a non-recurrent MFN trained with BP on the same data (Adamson and Damper, 1996). Alternatively, for German, Jordan networks have been shown to improve accuracy over BP's (Rosenke, 1995). Again, empirical tests will be needed to investigate in detail the potential advantages and drawbacks of this extension for neural networks and surely also for the decision-tree and instance-based learning methods.

Chapter 8

Conclusions

When the task of learning English word pronunciation is defined as learning to map fixed-size letter strings to phonemes with stress markers, it can be learned with adequate success by inductive-learning algorithms. Three of the tested algorithms were able to attain adequate accuracy: the two instance-based learning algorithms IB1 and IB1-IG, and the decision-tree algorithm IGTREE.

By defining word pronunciation as a mapping from fixed-size letter strings to phonemes with stress markers, i.e., as a one-pass classification task, we have avoided the explicit implementation of abstraction levels assumed necessary by mainstream linguistics to perform the task. Our demonstration of inductive-learning algorithms attaining adequate accuracy with the one-pass task definition implies that the linguistic abstraction levels are not needed explicitly in an adequately-performing word-pronunciation system (by our definition of adequate accuracy).

Sequential and parallel decomposition

Generalisation accuracy results with implementing linguistically-motivated decompositions of the word-pronunciation task, in Chapter 4, showed that a decomposition of the word-pronunciation task in five sequentially-coupled subtasks (as done in existing word-pronunciation systems) led to a worse generalisation accuracy than a decomposition of the task into three subtasks: it was demonstrated that the two pairs of graphemic parsing and grapheme-phoneme conversion, and syllabification and stress assignment, could best be integrated in single tasks when learned by the IGTREE learning algorithm under the adaptive variant.

Parallel decompositions of the word-pronunciation task were tested in Chapter 5, leading to the conclusion that grapheme-phoneme conversion and stress assignment, when learned independently by inductive-learning algorithms, perform better jointly than the sequential-modular systems investigated in Chapter 4. Two word-pronunciation systems with parallelised subtasks showed to be particularly well learnable; better than the best-performing sequential-modular system of Chapter 4: one system in which the two subtasks of grapheme-phoneme conversion are performed in parallel and one in which these two subtasks are defined as one single task, i.e., the one-pass mapping of instances of letter strings to phonemes with stress markers. The adequate accuracy of the former system, G/S, demonstrates that grapheme-phoneme conversion and stress assignment are fairly independent tasks. More importantly, the latter system, GS, proved the point that the word-pronunciation task can indeed be performed adequately when no linguistically-motivated abstraction level is assumed (even no explicit distinction between two parallel tasks) between those of letters and phonemes with stress markers.

In Chapter 6, the idea was elaborated that the one-pass word-pronunciation task GS could be learned better when the available training data was split according to a data-based criterion, and processed by a gating system. Minor improvements were indeed attained with two criteria, the first isolating a subset of assumed typical instances from the other instances, and the second isolating a subset of frequently-occurring instances. However, the tests failed to show that a data-based decomposition of the task could lead to significant accuracy improvements over those obtained with the non-decomposed task.

Exploring the minimisation of expert bias

While avoiding some abstraction levels assumed necessary by mainstream linguistic experts has proven profitable for the generalisation accuracy of inductively-learned word-pronunciation systems, the conclusion cannot be that linguistic theory is superfluous in building word-pronunciation systems. After all, a significant improvement over the best accuracy was obtained by constructing a system, G-S, which explicitly decomposes grapheme-phoneme conversion and stress assignment and places them in sequential order. This system, of which the subtasks were learned by the IB1-IG algorithm, demonstrates that explicit linguistic abstractions can indeed be proficient for obtaining high-accuracy performance given the CELEX data. The generalisation accuracy of this particular system, 3.14% incorrectly classified

phonemes, 3.05% incorrectly classified stress markers, and 33.79% incorrectly produced words, approaches high-quality accuracy demanded by industrial text-to-speech standards. Alternatively, two voting systems (GS-COMBINER and GS-ARBITER), which do not incorporate expert knowledge but combine the outputs of inductively-learned word-pronunciation systems to resolve, by voting over them, some of the errors made by these systems, are also successful in optimising generalisation accuracy. GS-ARBITER obtains the best overall performance reported in this thesis in terms of classified phonemes (2.88% incorrectly classified phonemes).

Discussions on attaining high-quality accuracy put aside, the main conclusion that can be drawn from the present study is that a reasonable amount of knowledge assumed necessary by mainstream linguistics and by developers of mainstream text-to-speech synthesis systems, is not needed explicitly for a word-phonemisation system to perform adequately. As far as abstractions are really needed for adequately performing word pronunciations, they can be learned, albeit implicitly and probably not in the form they are theoretically formulated, by induction from learned examples. The best-performing algorithms capable of producing adequate accuracy, IGTREE, IB1, and IB1, merely establish appropriate estimates of similarity between newly-encountered letter strings and stored instances of letter strings associated with pronunciation knowledge, gathered earlier in a learning phase. This leads us to conclude that the analogy principle as described in Chapter 1 can indeed be successfully implemented. The similarity function that is empirically demonstrated to be appropriate is based on two functions: one counting identical letters between two word-pronunciation instances, and one (in IGTREE and IB1-IG) assigning different weights (determined by information-theoretic estimates) to letter matches.

From a linguistic perspective our study has shown that it is a viable method of research to view the word-pronunciation task from the bottom up, i.e., to construct models of the task on the basis of the data itself, and investigate how far this bottom-line approach can reach. The ideas of De Saussure on the generic analogy principle and of Bloomfield on induction as a means to discover knowledge on language have been shown to be implementable, albeit in the small, restricted word-pronunciation task in the morpho-phonological domain. Assuming a complex hierarchy of abstraction levels for the task can be intuitive, insightful, and plausible, but may be a construct which enforces too much structure on the task that can be adequately learned and performed on the basis of very local knowledge (viz. a context of

seven letters) of one-level associations between (parts of) written words and their phonemic transcriptions. As far as abstraction levels are necessary for word pronunciation, they are largely implicit in the word-pronunciation data itself, and can as such be traced by learning algorithms to an adequate degree. Thus, there appears to be no need for an overly expert-theoretic bias in the definition of the word-pronunciation learning task when algorithms from supervised ML are the learners, since the dreaded variance in the absence of a bias can be avoided when a set of examples of the size of a dictionary is available. One should be careful extrapolating this apparent success to other language tasks, however. The word-pronunciation task is to map strings of letters to strings of phonemes and stress assignments, which is a relatively close mapping – the number of feature values and classes is small, and the many-to-many relations between them are calculable. When feature values represent words (of which there may be several hundreds of thousands), or when classes represent semantic representations, modularity would without a doubt be a prerequisite to reduce the complexity and the computational load of the overall learning task.

Lazy inductive language learning

From a machine-learning perspective we have shown that the investigated tasks do not allow for too much abstraction by forgetting when the goal is high generalisation accuracy. Applications of the lazy learning algorithms IB1 and IB1-IG to the word pronunciation task and subtasks consistently lead to the best accuracies; the more an algorithm attempts to abstract from the data by compressing it, the more generalisation accuracy is harmed (cf. Daelemans, 1996a). Moreover, we have found indications for two intrinsic characteristics of the data favouring information-gain-weighted lazy learning. First, the data contains many small disjuncts, i.e., clusters (families) of instances estimated to contain between about three to a hundred members sharing the same class. For learning small disjuncts, lazy learning is the preferred learning approach since eager learning tends to throw away essential information on how to reach the small disjuncts while lazy learning retains all instances in memory. Second, the feature values of word-pronunciation instances have varying relevances (the middle letter of a window being by far the most relevant for classification). By using this intrinsic characteristic of the data, IB1-IG skews the instance space in such a way that instances belonging to the same class are on average more similar to each other as compared to the similarity function used in IB1.

In general, our study suggests that abstraction by forgetting, pruning, and other implementations of the minimal description length principle (Rissanen, 1983) underlying many acclaimed machine-learning algorithms should be applied carefully when dealing with language tasks. The investigated task, word pronunciation, can only be mastered properly when the learning algorithm has the liberty and, metaphorically, the *patience* to learn about the hundred thousand-odd words, the few ten thousands of different morphemes that make up the words, and their different, sometimes noisy or ambiguous pronunciations. After all, a writing system of a language such as English is, by its morphemes and the way they can recombine, a method of describing practically anything in a compressed way – and compression halts somewhere near the level of morphemes.

Inductive language learning, a language-independent and data-oriented approach to natural language, opens the door to many exciting studies of different tasks, differently-sized corpora, in different languages – learning English word pronunciation is but one language task, and most likely not the most complex to be found. We hope that our demonstration that analogy between two levels of language can be exploited adequately by inductive learning is a step ahead in this still largely unknown territory.

Appendix A

Orthographic and phonemic alphabets

In this appendix we give the letter alphabet and the phoneme alphabet used in our experiments on English morpho-phonological (sub) tasks (Chapters 3, 4, 5, and 6). The letter alphabet is adapted from CELEX (Van der Wouden, 1990; Burnage, 1990) and contains 42 characters. These include the 26 letters of the roman alphabet, 12 letters with diacritics (mostly occurring in loan words), three non-letter characters which are treated by CELEX as letters, viz. the dot, the apostrophe, and the hyphen, and one character (_) denoting the space before and after a word. Table A.1 lists all 42 letters.

The phonemic alphabet is an adaptation of the DISC phoneme alphabet as proposed and used in CELEX (Burnage, 1990). It contains 62 phonemes, of

type	#	list
roman alphabet	26	**a b c d e f g h i j k l m n o p q r s t u v w x y z**
letters with diacritics	12	**à â ä ç è é ê ï ñ ô ö ü**
non-letter characters	3	. ' -
space	1	_

Table A.1: The contents of the letter alphabet used in the experiments with English morpho-phonology.

which 54 are directly extracted from CELEX, and eight are added to the phoneme alphabet representing eight combinations of two phonemes realised in spelling as one letter (e.g., the **x** in **taxi** is pronounced /ks/; in our phoneme alphabet, /ks/ is encoded as X). Table A.2 displays all 62 phonemes in their IPA (*International Phonetic Alphabet*, IPA, 1993) notation, their DISC notation (including the eight combined phonemes) used in our experiments, and for each phoneme one example word containing the phoneme in its pronunciation (indicated by underlined letters). The example words are adapted from the CELEX manual (Burnage, 1990).

Each phoneme is characterised uniquely by a combination of *articulatory features* present during articulation. We identify 25 articulatory features. The employed feature set is adapted from that of Sejnowski and Rosenberg (1987) and is also used by Dietterich *et al.* (1995). Table A.3 lists the names of the 25 articulatory features, and the numbers assigned to the features as used in the subsequent Tables. Tables A.4 and A.5 display for each phoneme its articulatory-feature vector. Consonant vowels are listed in Table A.4; vowel phonemes are listed in Table A.5. When an articulatory feature is marked with an asterisk (*) in a phoneme's row, it signifies that this articulatory feature is present during the pronunciation of that phoneme.

IPA	DISC	example	IPA	DISC	example	IPA	DISC	example
p	p	**pat**	h	h	**had**	ɔː	$	**born**
b	b	**bad**	w	w	**why**	uː	u	**soon**
t	t	**tack**	ʧ	J	**cheap**	ɜː	3	**burn**
d	d	**dad**	ʤ	-	**jeep**	eɪ	1	**bay**
k	k	**cad**	ˌŋ	C	**bacon**	aɪ	2	**buy**
g	g	**game**	ˌm	F	**idealism**	ɔɪ	4	**boy**
ŋ	N	**bang**	ˌn	H	**burden**	əʊ	5	**no**
m	m	**mad**	ˌl	P	**dangle**	aʊ	6	**browse**
n	n	**now**	*	R	**father**	ɪə	7	**peer**
l	l	**lad**	ks	X	**taxi**	ɛə	8	**pair**
r	r	**rat**	kʃ	+	**sexual**	ʊə	9	**poor**
f	f	**fat**	gz	G	**auxiliary**	æ̃	c	**timbre**
v	v	**vows**	ɪ	I	**pit**	ɑ̃ː	q	**détente**
ð	D	**that**	ɛ	E	**pet**	æ̃ː	0	**lingerie**
θ	T	**thin**	æ	{	**pat**	ɒ̃ː	~	**bouillon**
s	s	**sock**	ʌ	V	**putt**	aɪə	[	**admirer**
z	z	**zap**	ɒ	Q	**pot**	jʊ	}	**compute**
ʃ	S	**sheep**	ʊ	U	**put**	jə	]	**fabulous**
ʒ	Z	**measure**	ə	@	**another**	jɔː	>	**cure**
j	j	**yank**	iː	i	**bean**	jʊə	<	**annual**
χ	x	**loch**	ɑː	#	**barn**			

Table A.2: The contents of the phoneme alphabet used in the experiments with English morpho-phonology. For each phoneme, its IPA and DISC notation is given, as well as an example word illustrating its pronunciation.

number	name of articulatory feature
1	low
2	medium
3	high
4	tensed
5	back1
6	back2
7	central1
8	central2
9	front1
10	front2
11	voiced
12	labial
13	stop
14	velar
15	alveolar
16	unvoiced
17	fricative
18	glottal
19	glide
20	dental
21	liquid
22	nasal
23	palatal
24	affricative
25	elide

Table A.3: Names of the 25 articulatory features used for encoding the phonemes in the ART/S experiments (Section 5.3).

phoneme	articulatory feature																								
	low	medium	high	tensed	back1	back2	central1	central2	front1	front2	voiced	labial	stop	velar	alveolar	unvoiced	fricative	glottal	glide	dental	liquid	nasal	palatal	affricat.	elide
p												★	★			★									
b											★	★	★												
t													★		★	★									
k													★	★		★									
h																★		★	★						
w											★	★							★						
ɥ																★							★	★	
d											★		★		★										
ʤ											★			★								★			
g											★		★	★											
m											★									★			★		
ŋ											★			★								★			
n											★											★	★		
m̩											★	★										★			
l											★				★						★				
n̩											★			★								★			
*		★						★			★			★							★				
l̩											★									★	★				
ks							★			★						★								★	
r											★										★		★		
kʃ														★		★	★						★	★	
f												★				★	★								
gz											★			★			★							★	
v											★	★					★								
ð											★						★			★					
θ																★	★			★					
s															★	★	★								
z											★				★		★								
ʃ																★	★						★		
ʒ											★						★						★		
j											★								★				★		
χ											★			★											

Table A.4: Articulatory-feature vectors for all consonant phonemes. Each phoneme is characterised by a vector of 25 articulatory features. A star (★) in a cell denotes the presence of the articulatory feature in the respective column, in the pronunciation of the phoneme in the respective row.

phoneme	articulatory feature																								
	low	medium	high	tensed	back1	back2	central1	central2	front1	front2	voiced	labial	stop	velar	alveolar	unvoiced	fricative	glottal	glide	dental	liquid	nasal	palatal	affricative	elide
ɔː		★												★											
uː			★	★		★																			
3ː											★			★							★				
eɪ		★		★						★															
aɪ		★		★			★			★															
ɔɪ		★		★			★	★																	
əʊ		★		★		★																			
aʊ		★	★	★	★			★																	
ɪə		★	★					★	★																
ɛə		★						★	★	★															
ʊə		★	★		★			★																	
æ								★			★											★			
ɪ			★						★																
ɑː						★					★											★			
ɛ		★							★	★															
æː									★		★											★			
æ	★									★															
ɒː	★					★					★											★			
ʌ	★						★																		
aɪə		★	★				★	★	★																
ɒ		★			★									★											
jʊ			★	★			★		★	★									★				★		
ʊ			★		★																				
jə		★						★			★								★				★		
ə		★						★																	
jɔː	★					★					★								★				★		
iː			★	★					★																
jʊə		★	★		★						★								★				★		
ɑː	★			★				★																	

Table A.5: Articulatory-feature vectors for all vowel phonemes. Each phoneme is characterised by a vector of 25 articulatory features. A star (★) in a cell denotes the presence of the articulatory feature in the respective column, in the pronunciation of the phoneme in the respective row.

Appendix B

BP in MFN

The description of BP in MFNs given here is intended as a technical backup to the description given in Subsection 2.1.2. It is divided into two parts. First, we describe the feedforward activation of units in an MFN. Second, we describe how the error of units and the weight change of connections are computed when BP is employed. This description is based on Hertz, Krogh, and Palmer (1991).

For the sake of simplicity, we consider an MFN with an input layer (units indexed by i), one hidden layer (units indexed by j), and an output layer (units indexed by k).

The output of the network is defined as

$$o_k = f(e_k), \tag{B.1}$$

where o_k is the k-th output activation and $f()$ is a sigmoidal function defined as

$$f(x) = \frac{1}{1 + \exp^{-x}}. \tag{B.2}$$

The *net input* e_k of the k-th output unit is

$$e_k = \sum_j W_{kj} h_j + \theta_k, \tag{B.3}$$

where W_{kj} is the hidden-to-ouput weight from the j-th hidden to the k-th output unit, h_j is the output of the j-th *hidden* unit and θ_k is the *bias weight* of the k-th unit.

The hidden output is defined as

$$h_j = f(e_j) = f(\sum_i w_{ji} i_i + \theta_j). \tag{B.4}$$

Here, w_{ji} denotes the weight of the input-to-hidden connection from the i-th input to the j-th hidden unit, i_i is the i-th input value, and θ_j is the bias weight of the j-th unit.

Taken together, the above definitions lead to

$$o_k = f\left(\sum_j W_{kj} f\left(\sum_i w_{ji} i_i\right)\right). \tag{B.5}$$

Where we have omitted the bias weights which are treated as regular weights by assuming that they are connected to an additional unit with constant output 1.0.

The *error function* E specifies the classification error of the MFN on all pattern p, i.e.,

$$E = \sum_p E_p = \frac{1}{2}\sum_{pj}(t_{pk} - o_{pj})^2, \tag{B.6}$$

which can now be written as (introducing the pattern index p):

$$E = \frac{1}{2}\left[t_{pk} - f\left(\sum_j W_{kj} f\left(\sum_i w_{ji} i_{pi}\right)\right)\right]. \tag{B.7}$$

The weight-changing rule should perform a *gradient descent* on E. For the hidden-to-output weights this leads to

$$\begin{aligned} \Delta W_{kj} &= -\eta \frac{\partial E}{\partial W_{kj}} && \text{(B.8)} \\ &= \eta \sum_p (t_{pk} - o_{pk}) f'(e_{pk}) h_{pj} && \text{(B.9)} \\ &= \eta \sum_p \delta_{pk} h_{pj}, && \text{(B.10)} \end{aligned}$$

where η is the learning rate and δ_{pk} is the *output error signal* defined as

$$\delta_{pk} = f'(e_{pk})(t_{pk} - o_{pk}). \tag{B.11}$$

For the input-to-hidden weights, the error signals are not directly available and have to be derived from the output error signals δ_{pk}:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{B.12}$$

$$= -\eta \frac{\partial E}{\partial h_{pj}} \frac{\partial h_{pj}}{\partial w_{ji}} \tag{B.13}$$

$$= \eta \sum_{pk} (t_{pk} - o_{pk}) f'(e_{pk}) W_{kj} f'(e_{pj}) i_{pi} \tag{B.14}$$

$$= \eta \sum_{pk} \delta_{pk} W_{kj} f'(e_{pj}) i_{pi} \tag{B.15}$$

$$= \eta \sum_{p} \delta_{pj} i_{pi}, \tag{B.16}$$

where δ_{pj} is the *hidden error signal* defined as

$$\delta_{pj} = f'(e_{pj}) \sum_{k} W_{kj} \delta_{pk}. \tag{B.17}$$

Appendix C

Information gain

Information gain, a feature-weighting function, is used in IB1-IG (Subsection 2.1.3) and IGTREE (Subsection 2.1.4), to provide a real-valued expression of the relative importance of feature-values, i.e., letter positions, given a certain morpho-phonological task.

The idea behind computing the *information gain* of features is to interpret the training set as an information source capable of generating a number of messages (i.e., classifications) with a certain probability. The information entropy H of such an information source can be compared in turn for each of the features characterising the instances (let n equal the number of features), to the average information entropy of the information source when the value of those features are known.

Data-base information entropy $H(D)$ is equal to the number of bits of information needed to know the classification given an instance. It is computed by equation C.1, where p_i (the probability of classification i) is estimated by its relative frequency in the training set.

$$H(D) = -\sum_i p_i log_2 p_i \tag{C.1}$$

To determine the information gain of each of the n features $f_1 \ldots f_n$, we compute the average information entropy for each feature and subtract it from the information entropy of the data base. To compute the average information entropy for a feature f_i, given in equation C.2, we take the average information entropy of the data base restricted to each possible value for the feature. The expression $D_{[f_i=v_j]}$ refers to those patterns in the data base that have value v_j for feature f_i, j is the number of possible values of f_i, and V is the set of

possible values for feature f_i. Finally, $|D|$ is the number of patterns in the (sub) data base.

$$H(D_{[f_i]}) = \sum_{v_j \in V} H(D_{[f_i=v_j]}) \frac{|D_{[f_i=v_j]}|}{|D|} \tag{C.2}$$

Information gain of feature f_i is then obtained by equation C.3.

$$G(f_i) = H(D) - H(D_{[f_i]}) \tag{C.3}$$

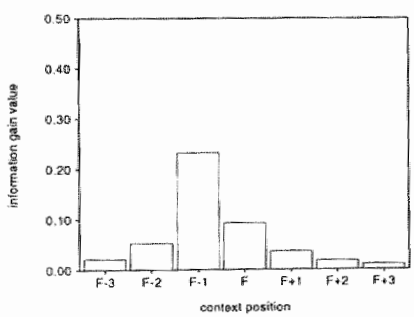
Appendix D

Information-gain values

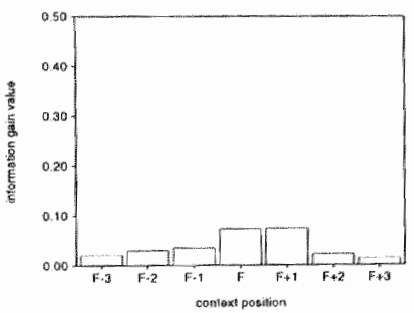
This appendix displays the information-gain values computed for the word-pronunciation (sub)tasks investigated in Chapters 3 and 5. The information gain of each of the seven input features is computed on the complete data base of the (sub)task. This information is presented in the form of bar graphs.

D.1 Isolated word-pronunciaton subtasks

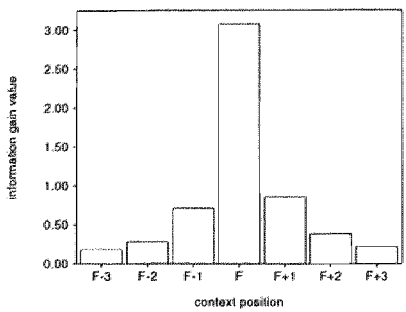
This section displays the information-gain contours for the word-phonemisation subtasks investigated in isolation in Chapter 3. The maximum value of the y-axis, viz. information gain, is held constant at 0.50 for comparison, except for the grapheme-phoneme conversion subtask. Feature positions are denoted by **F** with an offset determining the position from the focus letter.



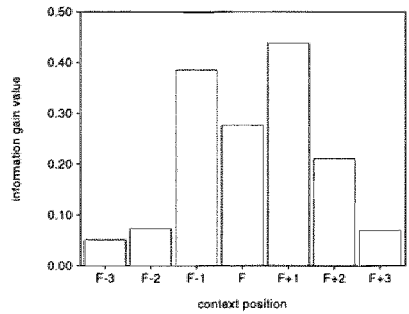
morphological segmentation
(Section 3.1)



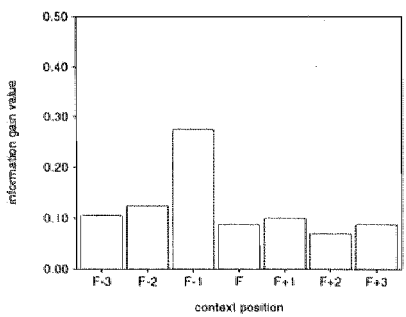
graphemic parsing
(Section 3.2)



grapheme-phoneme conversion
(Section 3.3)



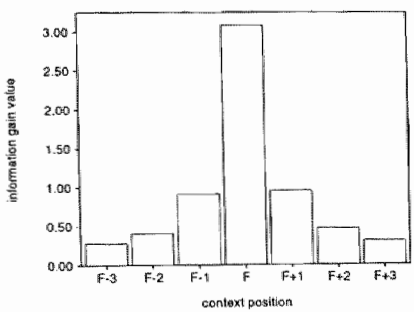
syllabification
(Section 3.4)



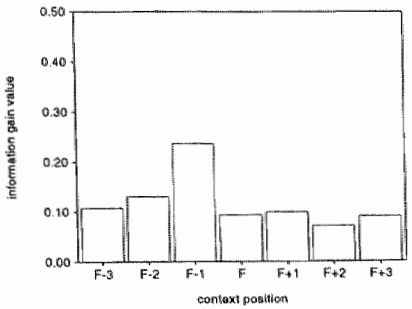
stress assignment
(Section 3.5)

D.2 Parallelised word-phonemisation

This section displays the information-gain contours for the word-pronunciation task and stress-assignment subtask investigated in Chapter 5.



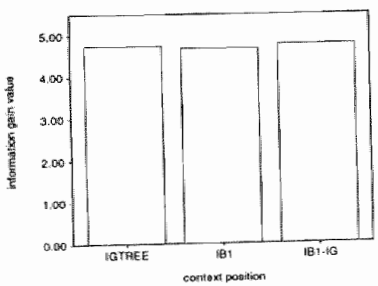
word phonemisation, GS (Section 5.1)



stress assignment in G/S and Art/S (Sections 5.2 and 5.3)

D.3 Voting

This section displays the information-gain values of the classification outputs of IGTREE, IB1, and IB1-IG, trained on the GS task, with the actual GS classifications as output (cf. Section 7.2).



Output of classifiers IGTREE, IB1, and IB1, trained on the GS task, with the actual GS classifications as output

Appendix E

IGTREE

We describe how IGTREE (Daelemans *et al.*, 1997a) performs the induction of decision trees in Section E.1, and describe how classifications are retrieved from induced decision trees in Section E.2.

E.1 Decision-tree induction by IGTREE

Given a training set of instances with n features, and given the information gain values of all features computed over the full corpus, a decision tree is built in which instances are stored in a compressed fashion (i.e., as partially overlapping paths of variable length):

Procedure **BUILD-IG-TREE**:

Input:

- A training set T of instances with their associated classifications (start value: a full instance base),
- an information-gain-ordered list of features $f_i \ldots f_n$ (start value: $f_1 \ldots f_n$).

Output: A subtree.

1. If T is unambiguous (all instances in T map to the same class c), or $i = (n + 1)$, create a leaf node with unique class label c.
2. Otherwise, until $i = n$ (the number of features)
   - Select the first feature f_i in $f_i \ldots f_n$, and construct a new node N for feature f_i, and as default class c (the class occurring most frequently in T).
   - Partition T into subsets $T_1 \ldots T_m$ according to the values $v_1 \ldots v_m$ which occur for f_i in T (instances with the same values for this feature in the same subset).

- For each $j \epsilon \{1, ..., m\}$:
  if not all instances in T_j map to class c, BUILD-IG-TREE (T_j, $f_{i+1} ... f_n$), connect the root of this subtree to N and label the arc with letter value v_j.

E.2 Decision-tree classification retrieval in IGTREE

After tree construction, the classification of an instance can be looked up in the tree. The instance is matched against paths in the decision tree, until a leaf node is found producing a unique class label or, if no unambiguous mapping can be found, producing the most probable class at the point of tree search failure:

Procedure **SEARCH-IG-TREE**:

Input:

- The root node N of an subtree (start value: top node of a complete IGTree),
- an unlabelled instance I with information-gain-ordered feature values $v_i ... v_n$ (start value: $v_1 ... v_n$).

Output: A class label.

1. If N is a leaf node, produce default class c associated with this node as output.
2. Otherwise, if no arc originates from N labelled with letter value v_i, then produce default class c associated with N as output.
3. Otherwise,
   - let M be the next node reached via the arc originating from N with as label letter v_i.
   - SEARCH-IG-TREE ($M, v_{i+1} ... v_n$)

References

Adamson M. and Damper R. (1996). A recurrent network that learns to pronounce English text. *Proceedings of the International Conference on Spoken Language Processing, ICSLP-96*, Vol. 4, pp. 1704–1707. (172)

Aha D. W. (1991). Incremental constructive induction: an instance-based approach. *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 117–121. Evanston, ILL: Morgan Kaufmann. (165)

Aha D. W. (1992). Generalizing from case studies: a case study. *Proceedings of the Ninth International Conference on Machine Learning*, pp. 1–10. San Mateo, CA: Morgan Kaufmann. (154, 155, 156)

Aha D. W. and Goldstone R. L. (1992). Concept learning and flexible weighting. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pp. 534–539. Bloomington, IN: Lawrence Erlbaum. (33)

Aha D. W., Kibler D., and Albert M. (1991). Instance-based learning algorithms. *Machine Learning*, Vol. 7, pp. 37–66. (14, 21, 25, 33, 34, 58, 153, 154)

Ali K. (1996). *Learning probabilistic relational concept descriptions.* PhD thesis, Department of Information and Computer Science, University of California at Irvine. (156)

Allen J., Hunnicutt S., and Klatt D. (1987). *From text to speech: The* MITalk *system.* Cambidge University Press, Cambridge, UK. (1, 42, 46, 48, 50, 60, 77, 82, 83, 148, 162, 168)

Anderson J. A. and Rosenfeld E. (eds.) (1988). *Neurocomputing: Foundations of research.* Cambridge, MA: The MIT Press. (28)

Andrews S. (1992). Frequency and neighborhood effects on lexical access: lexical similarity or orthographic redundancy? *Journal of Experimental Psychology: Learning, Memory and Cognition*, Vol. 18, pp. 234–254. (171)

Bauer L. (1983). *English word-formation*. Cambridge, UK: Cambridge University Press. (44)

Bauer L. (1988). *Introducing linguistic morphology*. Edinburgh, UK: Edinburgh University Press. (43)

Benedict P. (1990). The second data generation program – DGP/2. University of Illinois, Urbana-Champaign, Inductive Learning Group, Beckman Institute for Advanced Technology and Science. (155, 156)

Bird S. (1994). Introduction to Computational Phonology. *Computational Linguistics*, Vol. 20, No. 3, pp. 1–8. (168)

Bird S. (1996). *Computational phonology: a constraint-based approach*. Cambridge University Press, Cambridge, UK. (45, 168)

Blevins J. (1995). The syllable in phonological theory. *The handbook of phonological theory* (ed. J. A. Goldsmith), pp. 206–244. Cambridge, MA: Blackwell. (49)

Bloomfield L. (1933). *Language*. New York: Holt, Rinehard and Winston. (8, 43)

Breiman L. (1996a). Bagging predictors. *Machine Learning*, Vol. 24, No. 2. (150)

Breiman L. (1996b). Bias, variance and arcing classifiers. Technical Report 460, University of California, Statistics Department, Berkeley, CA. (150)

Breiman L., Friedman J., Ohlsen R., and Stone C. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth International Group. (14, 38, 153, 154, 155)

Brunak S. and Lautrup B. (1990). *Neural networks: Computers with intuition*. World Scientific, Singapore. (167)

Bullinaria J. (1993). Connectionist modelling of reading aloud. *Proceedings of The Cognitive Science of Natural Language Processing Workshop '93, Dublin*. (167)

Burnage G. (1990). CELEX: *A guide for users.* Centre for Lexical Information, Nijmegen. (26, 50, 179, 180)

Carbonell J. G. (1989). Introduction: Paradigms for machine learning. *Artificial Intelligence,* Vol. 40, pp. 1–9. (13)

Chan P. K. and Stolfo S. J. (1995). A comparative evaluation of voting and meta-learning of partitioned data. *Proceedings of the Twelfth International Conference on Machine Learning,* pp. 90–98. (134, 150, 151)

Chan P. K. and Stolfo S. J. (1997). Metrics for analysing the integration of multiple learned classifiers. submitted. (150)

Cheeseman P., Kelly J., Self M., Stutz J., Taylor W., and Freedman D. (1988). AUTOCLASS: a Bayesian classification system. *Proceedings of the Fifth International Machine Learning Conference,* pp. 54–64. Ann Arbor, MI: Morgan Kaufmann. (13)

Chomsky N. (1957). *Syntactic structures.* Den Haag: Mouton. (1, 9)

Chomsky N. (1965). *Aspects of the theory of syntax.* Cambridge, MA: The MIT Press. (9)

Chomsky N. (1975). *Reflections on language.* New York, NY: Pantheon Books. (9)

Chomsky N. (1995). *The minimalist program.* Cambridge, MA: The MIT Press. (9)

Chomsky N. and Halle M. (1968). *The sound pattern of English.* Harper and Row, New York, NY. (1, 10, 45, 49, 50, 168)

Clark P. and Niblett T. (1989). The CN2 rule induction algorithm. *Machine Learning,* Vol. 3, pp. 261–284. (20, 155)

Clements G. N. and Hume E. V. (1995). The internal organization of speech sounds. *The handbook of phonological theory* (ed. J. A. Goldsmith), pp. 245–306. Cambridge, MA: Blackwell. (48, 114)

Cohen W. W. (1995). Fast effective rule induction. *Proceedings of the Twelfth International Conference on Machine Learning,* pp. ??–??, Lake Tahoe, California. (20)

Coker C., Church K. W., and Liberman M. (1990). Morphology and rhyming: two powerful alternatives to letter-to-sound rules in speech synthesis. *Proceedings of the First* ESCA *Workshop on Speech Synthesis* (eds. G. Bailly and C. Benoît), pp. 83–86, Autrans, France. European Speech Communication Association. (46, 60, 77, 162)

Coltheart M. (1978). Lexical access in simple reading tasks. *Strategies of Information Processing* (ed. G. Underwood), pp. 151–216. London: Academic Press. (45, 46)

Coltheart M., Curtis B., Atkins P., and Halter M. (1993). Models of reading aloud: Dual-route and parallel-distributed-processing approaches. *Psychological Review*, Vol. 100, pp. 589–608. (167)

Coulmas F. (1989). *The writing systems of the world.* Oxford, UK: Blackwell Publishers. (3, 46, 48)

Cover T. M. and Hart P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, Vol. 13, pp. 21–27. (21, 33)

Cutler A. and Norris D. (1988). The role of strong syllables in segmentation for lexical access. *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 14, pp. 113–121. (50)

Daelemans W. (1987). *Studies in language technology: An object-oriented model of morphophonological aspects of Dutch.* PhD thesis, Katholieke Universiteit Leuven. (1, 46, 77, 82, 87, 114, 162)

Daelemans W. (1988). GRAFON: A grapheme-to-phoneme system for Dutch. *Proceedings Twelfth International Conference on Computational Linguistics (COLING-88), Budapest*, pp. 133–138. (46, 60, 82, 87, 114, 162)

Daelemans W. (1995). Memory-based lexical acquisition and processing. *Machine translation and the lexicon* (ed. P. Steffens), number 898 in Springer Lecture Notes in Artificial Intelligence, pp. 87–98. Berlin: Springer-Verlag. (5, 15, 16, 33, 159)

Daelemans W. (1996a). Abstraction considered harmful: lazy learning of language processing. *Proceedings of the Sixth Belgian–Dutch Conference on Machine Learning* (eds. H. J. Van den Herik and A. Weijters), pp. 3–12, Maastricht, The Netherlands. MATRIKS. (159, 176)

Daelemans W. (1996b). Experience-driven language acquisition and processing. *Proceedings of the CLS Opening Academic Year 1996-1997* (eds. M. Van der Avoird and C. Corsius), pp. 83–95. Tilburg: CLS. (2, 15, 159)

Daelemans W. and Van den Bosch A. (1992a). Generalisation performance of backpropagation learning on a syllabification task. *TWLT3: Connectionism and Natural Language Processing* (eds. M. F. J. Drossaers and A. Nijholt), pp. 27–37, Enschede. Twente University. (15, 25, 34, 57, 166)

Daelemans W. and Van den Bosch A. (1992b). A neural network for hyphenation. *Artificial Neural Networks 2* (eds. I. Aleksander and J. Taylor), Vol. 2, pp. 1647–1650, Amsterdam. North-Holland. (15)

Daelemans W. and Van den Bosch A. (1997). Language-independent data-oriented grapheme-to-phoneme conversion. *Progress in Speech Processing* (eds. J. P. H. Van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg), pp. 77–89. Berlin: Springer-Verlag. (64, 66, 68, 166)

Daelemans W., Gillis S., and Durieux G. (1994a). The acquisition of stress: a data-oriented approach. *Computational Linguistics*, Vol. 20, No. 3, pp. 421–451. (15, 169)

Daelemans W., Gillis S., and Durieux G. (1994b). Skousen's analogical modeling algorithm: A comparison with lazy learning. *Proceedings of the First NeMLaP Conference, Manchester, UK.* (25)

Daelemans W., Berck P., and Gillis S. (1996). Unsupervised discovery of phonological categories through supervised learning of morphological rules. *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pp. 95–100, Copenhagen, Denmark. (169)

Daelemans W., Van den Bosch A., and Weijters A. (1997a). IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, Vol. 11, pp. 407–423. (25, 34, 37, 38, 39, 57, 58, 79, 110, 142, 143, 195)

Daelemans W., Van den Bosch A., and Zavrel J. (1997b). A feature-relevance heuristic for indexing and compressing large case bases. *Poster Papers of the Ninth European Conference on Machine Learing* (eds. M. Van Someren and G. Widmer), pp. 29–38, Prague, Czech Republic. University of Economics. (160)

Daelemans W., Weijters A., and Van den Bosch A. (eds.) (1997c). *Workshop Notes of the ECML/MLnet familiarisation workshop on Empirical learning of natural language processing tasks*, Prague, Czech Republic. University of Economics. (166)

Danyluk A. P. and Provost F. J. (1993). Small disjuncts in action: learning to diagnose errors in the local loop of the telephone network 81. *Proceedings of the Tenth International Conference on Machine Learning*, pp. 81–88. San Mateo, CA: Morgan Kaufmann. (156)

De Saussure F. (1916). *Course de linguistique générale*. Paris: Payot. edited posthumously by C. Bally and A. Riedlinger. (2, 7, 8, 10, 45, 48)

Devijver P. .A. and Kittler J. (1982). *Pattern recognition. A statistical approach*. Prentice-Hall, London, UK. (33)

Derwing B. L. and Skousen R. (1989). Real time morphology: Symbolic rules or analogical networks. *Berkeley Linguistic Society*, Vol. 15, pp. 48–62. (10)

Dietterich T. G. and Bakiri G. (1991). Error-correcting output codes: A general method for improving multiclass inductive learning programs. *Proceedings of AAAI-91*, pp. 572–577. Menlo Park, CA. (136)

Dietterich T. G. and Bakiri G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, Vol. 2, pp. 263–286. (22, 106, 107)

Dietterich T. G., Hild H., and Bakiri G. (1995). A comparison of ID3 and Backpropagation for English text-to-speech mapping. *Machine Learning*, Vol. 19, No. 1, pp. 5–28. (22, 106, 113, 114, 137, 166, 167, 180)

Dresher E. and Kaye J. (1990). A computational learning model for metrical phonology. *Cognition*, Vol. 32, No. 2, pp. 137–195. (167, 169)

Ellison T. M. (1993). *Machine learning of phonological structure*. PhD thesis, University of Western Australia. (169)

Fahlman S. E. (1988). An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, Carnegie–Mellon University. (142)

Fahlman S. E. and Lebière C. (1990). The Cascade-correlation Learning Architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA. (142)

Faragó A. and Lugosi G. (1993). Strong universal consistency of neural network classifiers. *IEEE Transactions on Information Theory*, Vol. 39, pp. 1146–1151. (21)

Fisher D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, Vol. 2, pp. 139–172. (13)

Flach P. (1995). *Conjectures: an inquiry concerning the logic of induction.* PhD thesis, Katholieke Universiteit Brabant, Tilburg, The Netherlands. (5)

Freund Y. and Shapire R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (ed. L. Saitta), pp. 148–156. San Francisco, CA: Morgan Kaufmann. (150)

Friedman J. H., Kohavi R., and Yun Y. (1996). Lazy decision trees. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 717–724. Cambridge, MA: The MIT Press. (160)

Gardner H. (1987). *The mind's new science: A history of the cognitive revolution.* New York, NY: Basic Books, paperback edition. (8)

Gasser M. and Lee C.-D. (1990). Networks that learn about phonological feature persistance. *Connection Science*, Vol. 2, pp. 265–278. (169)

Gillis S., Durieux G., Daelemans W., and Van den Bosch A. (1993). Learnability and markedness: Dutch stress assignment. *Proceedings of the 15th Conference of the Cognitive Science Society 1993, Boulder, CO*, pp. 452–457. (15)

Gillis S., Durieux G., and Daelemans W. (1995). A computational model of P&P: Dresher and Kaye (1990) revisited. *Approaches to parameter setting* (eds. M. Verrips and F. Wijnen), Vol. 4 of *Amsterdam Studies in Child Langage Development*, pp. 135–173. (169)

Glushko R. J. (1979). The organisation and activation of orthographic knowledge in reading aloud. *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 5, pp. 647–691. (2, 45)

Golding A. R. (1991). *Pronouncing names by a combination of rule-based and case-based reasoning.* PhD thesis, Stanford University, Stanford, CA. (136)

Goldsmith J. (1976). An overview of autosegmental phonology. *Linguistic Analysis*, Vol. 2, pp. 23–68. (45, 168)

Goldsmith J. A. (1995). Phonological theory. *The handbook of phonological theory* (ed. J. A. Goldsmith), pp. 1–23. Cambridge, MA: Blackwell. (49)

Goldsmith J. A. (ed.) (1996). *The handbook of phonological theory.* Cambridge, MA: Blackwell Publishers. (45, 59)

Gupta P. and Touretzky D. (1992). A connectionist learning approach to analysing linguistic stress. *Advances in Neural Information Processing Systems* (eds. J. Moody, S. J. Hanson, and R. P. Lippmann), Vol. 4, pp. 225–234. San Mateo, CA: Morgan Kaufmann. (167, 169)

Halle M. (1978). Knowledge unlearned and untaught: what speakers know about the sounds of their language. *Linguistic theories and psychological reality* (eds. M. Halle, J. W. Bresnan, and G. A. Miller), pp. 294–303. Cambridge, MA: The MIT Press. (114)

Halle M. and Keyser S. J. (1971). *English stress: its form, its growth, and its role in verse.* Harper and Row, New York, NY. (83)

Harnad S. (1982). Metaphor and mental duality. *Language, mind, and brain* (eds. T. Simon and R. Scholes), pp. 189–211. Hillsdale, NJ: Lawrence Erlbaum. (5)

Harris R. (1987). *Reading Saussure.* La Salle, IL: Open Court. (7, 10)

Hays W. L. (1988). *Statistics.* Orlando, FA: Holt, Rinehart and Winston, fourth edition. (55, 56, 62)

Hertz J., Krogh A., and Palmer R. G. (1991). *Introduction to the theory of neural computation.* Reading, MA: Addison-Wesley. (185)

Holte R. C., Acker L. E., and Porter B. W. (1989). Concept learning and the problem of small disjuncts. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 813–818. San Mateo, CA: Morgan Kaufmann. (40, 122, 156, 159, 164)

Hornik K., Stinchcombe M., and White H. (1989). Multilayer feedforward networks are universal approximators. Technical report, Department of Economics, UCSD, San Diego, CA. (21)

Hunnicutt S. (1976). Phonological rules for a text-to-speech system. *American Journal of Computational Linguistics*, Vol. Microfiche 57, pp. 1–72. (82, 83)

Hunnicutt S. (1980). Grapheme-phoneme rules: a review. Technical Report STL QPSR 2-3, Speech Transmission Laboratory, KTH, Sweden. (46, 82, 83)

Hunt E. B. (1962). *Concept learning: an information processing problem*. New York: Wiley. (21)

Hunt E. B., Marin J., and Stone P. J. (1966). *Experiments in induction*. New York, NY: Academic Press. (21, 38)

IPA (1993). Extensions to IPA alphabet. Technical report, International Phonetic Association.

Ivanova I. and Kubat M (1995). Initialization of neural networks by means of decision trees. *Knowledge Based Systems, Special Issue on Knowledge Based Neural Networks*, Vol. 8, No. 6. (23)

Jacobs R. A., Jordan M. I., Nowlan S. J., and Hinton G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, Vol. 3, pp. 79–87. (121)

Jared D., McRae K., and Seidenberg M.S. (1990). The basis of consistency effects in word naming. *Journal of Memory and Language*, Vol. 29, pp. 687–715. (46)

John G., Kohavi R., and Pfleger K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 121–129. San Mateo, CA: Morgan Kaufmann. (35, 36)

Jones D., Gimson A. C., and Ramsaran S. (1991). *English pronouncing dictionary*. Cambridge : Cambridge University Press, 14 edition. (52)

Jordan M. I. (1986). Attractor dynamics and parallellism in a connectionist sequential machine. *Proceedings of the Eighth Annual Meeting of the*

Cognitive Science Society, pp. 531–546. Hillsdale, NJ: Lawrence Erlbaum Associates. (172)

Kager R. (1995). The metrical theory of word stress. *The handbook of phonological theory* (ed. J. A. Goldsmith), pp. 367–402. Cambridge, MA: Blackwell. (50)

Katz L. and Frost R. (1992). The reading process is different for different orthographies: the orthographic depth hypothesis. *Haskins Laboratories Status Report on Speech Research 1992*, pp. 147–160. Haskins Laboratories. (48)

Kenstowicz M. (1993). *Phonology in generative grammar.* Oxford, UK: Basil Blackwell. (10, 45, 46, 48, 49, 59, 101)

Kibler D. and Aha D. W. (1987). Learning representative exemplars of concepts: an initial case study. *Proceedings of the Fourth International Workshop on Machine Learning* (ed. P. Langley), pp. 24–30. San Mateo, CA: Morgan Kaufmann. (14)

Kiparski P. (1995). The phonological basis of sound change. *The handbook of phonological theory* (ed. J. A. Goldsmith), pp. 640–670. Cambridge, MA: Blackwell. (117)

Knuth D. (1973). *The art of computer programming*, Vol. 3. Reading, MA: Addison-Wesley. (38)

Kodratoff Y. and Michalski R. (eds.) (1990). *Machine learning: An artificial intelligence approach*, Vol. III. San Mateo, CA: Morgan Kaufmann. (13)

Kolodner J. (1993). *Case-based reasoning.* San Mateo, CA: Morgan Kaufmann. (21, 33)

Koskenniemi K. (1984). A general computational model for wordform recognition and production. *Proceedings of the Tenth International Conference on Computational Linguistics / 22nd Annual Conference of the Association for Computational Linguistics*, pp. 178–181. (44, 168)

Koskenniemi K. and Church K. W. (1988). Complexity, two-level morphology and Finnish. *Proceedings of the Twelfth International Conference on Computational Linguistics*, pp. 335–340. John von Neumann Society for Computing Sciences. (45, 168)

Langley P. (1996). *Elements of machine learning*. San Mateo, CA: Morgan Kaufmann. (13)

Lavrac N. and Džeroski S. (1994). *Inductive logic programming*. Chichester, UK: Ellis Horwood. (21)

Lavrac N., Weber I., Zupanič D., Kazakov D., Štěpánková ., and Džeroski S. (1996). ILPNET repositories on WWW: Inductive logic programming systems, datasets and bibliography. *AI Communications*, Vol. 9, No. 4, pp. 157–206. (21)

Lawrence S., Giles C. L., and Tsoi A. C. (1996). What size neural network gives optimal generalization? Convergence properties of backpropagation. Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. (30)

Lehnert W. (1987). Case-based problem solving with a large knowledge base of learned cases. *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pp. 301–306. Los Altos, CA: Morgan Kaufmann. (166)

Lepage Y. and Shin-ichi A. (1996). Saussurian analogy: a theoretical account and its application. *Proceedings of COLING-96*, pp. 717–722, Copenhagen, Denmark. (2)

Levelt W. J. M. (1989). *Speaking: From intention to articulation*. Cambridge, MA: The MIT Press. (45, 46, 59)

Liberman M. and Prince A. (1977). On stress and linguistic rhythm. *Linguistic Inquiry*, , No. 8, pp. 249–336. (45, 168)

Ling C. X. (1994). Learning the past tense of English verbs: The symbolic pattern associator vs. connectionist models. *Journal of Artificial Intelligence Research*, Vol. 1, pp. 209–229. (167)

Matthews P. H. (1974). *Morphology*. Cambridge, UK: Cambridge University Press. (43)

Michalski R. (1993). Toward a unified theory of learning: multistrategy task-adaptive learning. *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems* (eds.

B. G. Buchanan and D. C. Wilkins), pp. 7–38. San Mateo, CA: Morgan Kaufmann. (13)

Michalski R. and Tecuci G. (eds.) (1994). *Machine learning: A multistrategy approach*, Vol. IV. Cambridge, MA: The MIT Press. (13)

Michalski R. S., Carbonell J. G., and Mitchell T. M. (eds.) (1983). *Machine learning: An artificial intelligence approach*, Vol. I. Palo Alto, CA: Tioga Publishing Company. (13)

Michalski R. S., Carbonell J. G., and Mitchell T. M. (eds.) (1986). *Machine learning: An artificial intelligence approach*, Vol. II. San Mateo, CA: Morgan Kaufmann. (13)

Mitchell T. (1997). *Machine learning*. New York, NY: McGraw Hill. (13)

Mohanan K. P. (1986). *The theory of lexical phonology*. Dordrecht: D. Reidel. (45)

Moody J. (1992). The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. *Advances in Neural Information Processing Systems* (eds. J. Moody, S. J. Hanson, and R. P. Lippmann), Vol. 4, pp. 847–854. San Mateo, CA: Morgan Kaufmann. (30)

Murphy P. and Aha D. W. (1995). UCI repository of machine learning databases – a machine-readable repository. Maintained at the Department of Information and Computer Science, University of California, Irvine. Anonymous ftp from `ics.uci.edu` in the directory `pub/machine-learning/databases`. (56, 166)

Norris D. (1993). A quantitative model of reading aloud. Technical report, MRC Applied Psychology Unit, Cambridge, UK. (167)

Oflazer K. and Somers H. (eds.) (1996). *Proceedings of the Second International Conference on New Methods in Language Processing*. Bilkent University, Ankara, Turkey. (165)

Parker D. B. (1985). Learning logic. Technical report, Center for Computational Research in Economics and Management Science, M.I.T. (28)

Piatelli–Palmarini M. (ed.) (1980). *Language learning: The debate between Jean Piaget and Noam Chomsky*. Cambridge, MA: Harvard University Press. (1, 7, 9)

Plaut D. C., McClelland J. L., Seidenberg M. S., and Patterson K. (1996). Understanding normal and impaired word reading: computational principles in quasi-regular domains. *Psychological Review*, Vol. 103, No. 1, pp. 56–115. (46, 167)

Powers D. and Daelemans W. (1991). SHOE: the extraction of hierarchical structure for machine learning of natural language. ITK Memo 10, ITK, Tilburg University. (165)

Prechelt L. (1994). Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 19/94, Fakultät für Informatik, Universität Karlsruhe, Germany. (56)

Provost F. J. and Aronis J. M. (1996). Scaling up inductive learning with massive parallelism. *Machine Learning*, Vol. 23, No. 1, p. 33. (156)

Quinlan J. R. (1986). Induction of decision trees. *Machine Learning*, Vol. 1, pp. 81–206. (14, 21, 35, 113, 137)

Quinlan J. R. (1993). C4.5: *Programs for machine learning*. San Mateo, CA: Morgan Kaufmann. (20, 25, 35, 37, 38, 39, 40, 41, 58, 83, 159)

Rissanen J. (1983). A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, Vol. 11, pp. 416–431. (20, 177)

Robins R. H. (1997). *A short history of linguistics*. London: Longman, 4 edition. (2, 8)

Röhr H. M. (1994). *Writing: Its evolution and relation to speech*. Bochum Publications in Evolutionary Cultural Semiotics. Bochum: Universitätsverlag Dr Norbert Brockmeyer. (4, 46, 48)

Rosch E. and Mervis C. B. (1975). Family resemblances: studies in the internal structure of categories. *Cognitive Psychology*, Vol. 7, pp. ??–?? (126)

Rosenblatt F. (1958). The perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review*, Vol. 65, pp. 368–408. (21)

Rosenke K. (1995). Verschiedene neuronale Strukturen für die Transkription von deutschen Wörtern. *Elektronische Sprachsignalverarbeitung, 6. Konferenz* (eds. R. Hoffman and R. Ose), pp. 159–166. Dresden: Institut für Technische Akustik. (172)

Rumelhart D. E. and McClelland J. L. (eds.) (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1: Foundations. Cambridge, MA: The MIT Press. (28)

Rumelhart D. E., Hinton G. E., and Williams R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (eds. D. E. Rumelhart and J. L. McClelland), Vol. 1: Foundations, pp. 318–362. Cambridge, MA: The MIT Press. (22, 25, 28, 29, 58)

Safavian S. R. and Landgrebe D. A. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, pp. 660–674. (38)

Salzberg S. L. (1995). On comparing classifiers: a critique of current reseach and methods. Technical Report JHU-95/06, Johns Hopkins University. (56, 57, 149)

Sampson G. (1984). *Writing systems: a linguistic introduction.* London: Hutchinson. (46)

Sarle W. (1997). Neural Networks Frequently Asked Questions. URL: `ftp://ftp.sas.com/pub/neural/FAQ.html`. (30)

Schaffer C. (1993). Overfitting avoidance as bias. *Machine Learning*, Vol. 10, No. 2, pp. xx–xx. (14)

Schaffer C. (1994). A conservation law for generalization performance. *Proceedings of the Eleventh International Machine Learning Conference* (eds. W. W. Cohen and H. Hirsch), pp. 259–265. Rutgers University, New Brunswick, NJ. (14)

Sejnowski T. J. and Rosenberg C. S. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, Vol. 1, pp. 145–168. (22, 66, 91, 107, 113, 123, 137, 166, 180)

Selkirk E. O. (1984). On the major class features and syllable theory. *Language Sound Structure* (eds. M. Aronoff and R. T. Oehrle), pp. 107–136. Cambridge, MA: The MIT Press. (49, 101)

Sharkey A. J. C. (1997). Modularity, combining and artificial neural nets. *Connection Science, Special Issue on Combining Artificial Neural Nets: Modular Approaches*, Vol. 9, No. 1, pp. 3–10. (150)

Shavlik J. W. and Dietterich T. G. (eds.) (1990). *Readings in Machine Learning*. San Mateo, CA: Morgan Kaufmann. (5, 12, 13)

Simon H. A. (1983). Search and reasoning in problem-solving. *Artificial Intelligence*, Vol. 21, pp. 7–29. (13)

Skousen R. (1989). *Analogical modeling of language*. Dordrecht: Kluwer Academic Publishers. (10, 11, 15, 20, 25)

Sproat R. (1992). *Morphology and computation*. ACL-MIT Press Series in Natural Language Processing. Cambridge, MA: The MIT Press. (10, 43, 44, 45, 168)

Stanfill C. and Waltz D. (1986). Toward memory-based reasoning. *Communications of the* ACM, Vol. 29, No. 12, pp. 1213–1228. (33, 137, 166)

Ting K. M. and Low B. T. (1997). Model combination in the multiple-data-batches scenario. *Machine Learning: Proceedings of ECML-97* (eds. M. Van Someren and G. Widmer), number 1224 in Lecture Notes in Artificial Intelligence, pp. 250–265. Berlin: Springer-Verlag. (150)

Tranel B. (1995). Current issues in French phonology. *The handbook of phonological theory* (ed. J. A. Goldsmith), pp. 798–816. Cambridge, MA: Blackwell. (66)

Treiman R. and Zukowski A. (1990). Toward an understanding of English syllabification. *Journal of Memory and Language*, Vol. 29, pp. 66–85. (50, 71, 78)

Van den Bosch A. and Daelemans W. (1992). Linguistic pattern matching capabilities of connectionist networks. *Proceedings of the Computational Linguistics in the Netherlands meeting 1991* (eds. J. van Eijk and W. Meyer), pp. 40–53. Utrecht: OTS. (22, 166)

Van den Bosch A. and Daelemans W. (1993). Data-Oriented Methods for Grapheme-to-Phoneme Conversion. *Proceedings of the 6th Conference of the EACL*, pp. 45–53. (15, 22, 166)

Van den Bosch A., Content A., Daelemans W., and De Gelder B. (1995). Measuring the complexity of writing systems. *Journal of Quantitative Linguistics*, Vol. 1, No. 3. (15, 22, 47, 48, 166)

Van den Bosch A., Weijters A., Van den Herik H. J., and Daelemans W. (1995b). The profit of learning exceptions. *Proceedings of the 5th Belgian-Dutch Conference on Machine Learning*, pp. 118–126. (22, 57)

Van den Bosch A., Daelemans W., and Weijters A. (1996). Morphological analysis as classification: an inductive-learning approach. *Proceedings of the Second International Conference on New Methods in Natural Language Processing, NeMLaP-2, Ankara, Turkey* (eds. K. Oflazer and H. Somers), pp. 79–89. (22)

Van der Wouden T. (1990). Celex: Building a multifunctional polytheoretical lexical data base. *Proceedings of BudaLex'88* (ed. T. Magay). Budapest: Akadémiai Kladó. (26, 50, 179)

Van Heuven V. J. and Pols L. C. W. (1993). *Analysis and synthesis of speech, strategic research towards high-quality text-to-speech generation.* Berlin: Mouton de Gruyter. (77)

Venezky R. L. (1970). *The structure of English orthography.* The Hague: Mouton. (46, 48)

Voisin J. and Devijver P. A. (1987). An application of the Multiedit-Condensing technique to the reference selection problem in a print recognition system. *Pattern Recognition*, Vol. 5, pp. 465–474. (21)

Vroomen J. and Van den Bosch A. (to appea). A Connectionist Model for Bootstrap Learning of Syllabic Structure. To appear in Language and Cognitive Processes. (162)

Weigend A. S. and Rumelhart D. E. (1994). Weight elimination and effective network size. *Computational Learning Theory and Natural Learning Systems* (eds. S. J. Hanson, G. A. Drastal, and R. L. Rivest), Vol. 1: Constraints and Prospects, chapter 16, pp. 457–476. MIT Press, Cambridge, MA. (30)

Weijters A. (1990). NetSpraak: een spraakmakend back-propagation netwerk. *Proceedings van den Derde Nederlandstalige AI Conferentie, NAIC-90* (eds. H. J. Van den Herik and N. Mars), pp. 137–146. (172)

Weijters A. (1991). A simple look-up procedure superior to NETtalk? *Proceedings of the International Conference on Artificial Neural Networks - ICANN-91, Espoo, Finland.* (15, 22, 166)

Weijters A. and Hoppenbrouwers G. (1990). NetSpraak: Een Neuraal Netwerk voor Grafeem-Foneem-Omzetting. *TABU*, Vol. 20, No. 1, pp. 1–25. (171)

Weiss S. and Kulikowski C. (1991). *Computer systems that learn.* San Mateo, CA: Morgan Kaufmann. (54, 55, 157, 162)

Werbos P. J. (1974). *Beyond regression: new tools for the prediction and analysis in the behavioral sciences.* PhD thesis, Harvard University. (28)

Wettschereck D. (1995). *A study of distance-based machine-learning algorithms.* PhD thesis, Oregon State University. (35)

Wettschereck D., Aha D. W., and Mohri T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, Vol. 11, pp. 273–314. (35, 36, 155)

White H. (1990). Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, Vol. 3, pp. 535–550. (21)

Wilson D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics*, Vol. 2, pp. 408–421. (21, 122)

Wolpert D. H. (1990). Constructing a generalizer superior to NETtalk via a mathematical theory of generalization. *Neural Networks*, Vol. 3, pp. 445–452. (166)

Wolters M. (1996). A dual route neural net approach to grapheme-to-phoneme conversion. *Artificial neural networks* – ICANN 96 (ed. C. Van der Malsburg *et al.*), pp. 233–238. Berlin: Springer Verlag. (167)

Wolters M. (1997). A diphone-based text-to-speech system for Scottish Gaelic. Master's thesis, Department of Computer Science, Universität Bonn. (48, 106, 113, 166)

Wolters M. and Van den Bosch A. (1997). Automatic phonetic transcription of words based on sparse data. *Workshop notes of the ECML/MLnet Familiarization Workshop on Empirical Learning of Natural Language Processing Tasks* (eds. W. Daelemans, A. Van den Bosch, and A. Weijters), pp. 61–70, Prague, Czech Republic. University of Economics. (166, 167)

Yvon F. (1996). *Prononcer par analogie: motivation, formalisation et évaluation.* PhD thesis, Ecole Nationale Supérieure des Télécommunication, Paris. (2, 57, 69, 123, 136, 137, 166, 167)

Zhang J. (1992). Selecting typical instances in instance-based learning. *Proceedings of the International Machine Learning Conference 1992*, pp. 470–479. (122, 125, 127, 134)

Zipf G. K. (1935). *The psycho-biology of language: an introduction to dynamic philology.* Cambridge, MA: The MIT Press. Second paperback edition, 1968. (171)

Index

Summary

Learning to pronounce written words means learning the intricate relations between the speech sounds of a language and its spelling. For languages with alphabetic writing systems, such as English, the relations can be captured to a large extent by *induction* (reasoning by analogy). After all, a pervasive phenomenon in alphabetic writing systems is that similarly-spelled words have similar pronunciations. However, mainstream (Chomskyan) linguistic theories have put forward the claim that pronouncing known and unknown words cannot be performed without the assumption of several levels of abstraction between spelling and pronunciation. Since general-purpose inductive-learning methods cannot discover such abstraction levels autonomously, linguistic theorists claim that inductive-learning methods cannot learn to pronounce words as well as generalise this knowledge to previously unseen words.

The present study challenges this claim. The study is embedded in both (i) the tradition of *structural* linguistics, building quite directly on ideas expressed a century ago by De Saussure, and (ii) the recent developments in machine learning (a subdomain of artificial intelligence). De Saussure claimed that language processing can be performed by assuming only two basic operations: segmentation and classification. In the machine-learning domain, it is claimed, and occasionally demonstrated, that inductive learning methods can learn complex real-world segmentation and classification tasks, attaining a high level of generalisation accuracy when given a sufficient amount of examples. The apparent and intriguing contrast between the claims from mainstream (Chomskyan) linguistics on the one hand, and those of structural linguistics and machine learning on the other hand, prompted us to perform an empirical study of the inductive learning of word pronunciation. The results of this empirical study allow us to claim that *inductive-learning algorithms can learn to pronounce written words with adequate generalisation accuracy, even*

when the task definition (and thus the system architecture) does not reflect explicitly any of the levels of abstraction assumed necessary by linguistic theories.

Chapter 1 introduces the historical background mentioned above on arguments and counterarguments concerning the feasibility of inductive language learning. On the basis of claims from structural linguistics and machine learning, the problem statement is formulated.

Chapter 2 provides the reader with an overview of relevant background knowledge. It describes three groups of inductive-learning algorithms: connectionist learning, instance-based learning, and decision-tree learning. They are suited, in principle, for learning word pronunciation. The chapter reviews mainstream linguistic views on the domains of morphology and phonology, highlighting levels of abstraction assumed present in word pronunciation. It then introduces the resource of word-pronunciation examples used in our study, i.e., the CELEX English lexical data base, and describes the general experimental methodology employed throughout the study.

Chapter 3 presents the application of the selected learning algorithms to five subtasks of the word-phonemisation task: (i) morphological segmentation, (ii) graphemic parsing, (iii) grapheme-phoneme conversion, (iv) syllabification, and (v) stress assignment. They represent five linguistically-motivated abstraction levels of word pronunciation. The results obtained with the five algorithms trained on each of the five subtasks in isolation indicate that the learning algorithms attain reasonable to excellent generalisation accuracy. Moreover, the results indicate that the less a learning algorithm abstracts from the learning material by data compression, the better its generalisation accuracy is on any of the subtasks.

In Chapter 4, modular word-pronunciation systems are constructed, learned, and tested. The architecture of the modular systems is inspired by two existing text-to-speech systems. Both modular systems perform the five subtasks investigated in Chapter 3. Each subtask is assigned to a single module; the five modules perform their subtasks in sequence. Generalisation-accuracy results indicate that cascading errors passed on between the modules seriously impede the overall accuracy of the systems. To prevent some of the errors we abandon the assumption that a five-modular decomposition is necessary, and investigate two three-modular systems, in which two pairs of subtasks from the five-modular systems are integrated into single tasks. The systems distinguish between (i) morphological segmentation, (ii) grapheme-phoneme conversion, and (iii) stress assignment. Their generalisation accuracy is significantly better than that of their five-modular counterparts.

In Chapter 5 the concept of sequential modularisation is abandoned and the alternative of parallel modularisation is tested in three new modular systems. In the first system, word pronunciation is performed by a single module converting spelling to phonemic transcriptions with stress markers in a single classification pass. The second system performs two tasks in parallel, viz. the conversion of letters to phonemes and the conversion of letters to stress markers. In the third system, the letter-phoneme conversion task is split further into 25 partial subtasks: each of these subtasks represents the detection of one articulatory feature of the phoneme to be classified. The results indicate that the single-module and two-module parallel systems perform better, and the articulatory-feature-detection system performs worse than the best three-module sequential system described in Chapter 4, when trained with the same algorithms.

Chapter 6 deals with three linguistically-uninformed gating systems for word pronunciation. In these systems, the word-pronunciation task is split in two parallel-processed partial word-pronunciation tasks. Rather than decomposing the task on the output level, the task is decomposed by applying a gating criterion at the input level, viz. on the spelling of (parts of) words. Three gating systems are tested: randomised gating, typicality-based gating, and occurrence-based gating. Randomised gating is demonstrated to be learned with lower accuracy than the word-pronunciation task as a whole; the typicality-based and occurrence-based systems are found to perform as accurate as the undecomposed system. Thus, gating does not lead to improvements in generalisation accuracy, but is capable of automatically decomposing the word-pronunciation data in essentially different subsets.

A summary of results reported in Chapters 3 to 6 is given in Chapter 7. Additional attention is paid to measures of computational efficiency of the learning algorithms tested. Instance-based learning attains the best (and adequate) generalisation accuracy on all (sub)tasks; the systems induced by decision-tree learning provide the best trade-off between generalisation accuracy and computational efficiency. Furthermore, analyses on the word-pronunciation data are performed searching for the cause of the success of instance-based learning: the analyses indicate that instances of word pronunciation tend to come in families containing small amounts of identically-classified members. Instance-based learning performs favourably with data containing this type of instance families (small disjuncts). Subsequently, the chapter describes two modular systems that combine apparently successful attributes of different systems investigated throughout the study, showing that better systems can

be built on the basis of both linguistic and empirical findings. The chapter then summarises related research, gives an overview of the limitations of the present approach, and indicates topics of future research.

In Chapter 8 the conclusion is drawn that inductive-learning methods, specifically instance-based learning algorithms, can learn the task of word phonemisation, attaining an adequately high level of generalisation accuracy. Linguistic bias in the task definition (and in the system architecture) can be reduced to an absolute minimum, thus be left *implicit*, while the system still attains accurate generalisation.

Samenvatting

Het leren van de uitspraak van woorden is het ontdekken van de ingewikkelde relaties tussen de klanken van een taal en haar spelling. In het geval van talen met een alfabetisch schrijfsysteem, zoals het Engels en het Nederlands, kunnen deze relaties voor een groot gedeelte worden gevonden door middel van *inductie* (redeneren door analogie). Voor alfabetische schrijfsystemen geldt immers dat woorden die in hun spelling op elkaar lijken, ook in hun uitspraak op elkaar lijken. De belangrijkste taalkundige theorieën (die voortbouwen op ideeën van Chomsky) beweren echter dat het uitspreken van bekende en onbekende woorden niet mogelijk is zonder aan te nemen dat er verschillende abstractieniveaus bestaan tussen spelling en uitspraak. Taaltheoretici stellen dat inductief-lerende methoden niet in staat zijn om woorduitspraak te leren en om de geleerde kennis toe te passen op nieuwe, onbekende woorden, omdat inductief-lerende methoden niet in staat zijn om uit zichzelf dergelijke abstractieniveaus te ontdekken.

Dit proefschrift zet een vraagteken bij deze stelling. De studie is ingebed in (i) de traditie van de *structurele* taalkunde, die vrij rechtstreeks voortbouwt op de ideeën van De Saussure van een eeuw geleden, en in recente ontwikkelingen binnen (ii) automatisch leren (*machine learning*), een deelgebied van de kunstmatige intelligentie. De Saussure stelde dat het verwerken van taal mogelijk is onder de aanname van slechts twee operaties: segmentatie en classificatie. Binnen het automatisch leren geldt de opvatting dat inductieve leermethoden in staat zijn om complexe, reële segmentatie- en classificatietaken te leren, mits er voldoende leervoorbeelden voorhanden zijn; er bestaan verschillende voorbeelden van toepassingen die dit aantonen. De in het oog lopende en intrigerende tegenstelling tussen de stellingen van de Chomskyaanse taalkunde aan de ene kant, en die van de structurele taalkunde en het automatisch leren aan de andere kant, bracht ons tot het uitvoeren van een empirische studie van het inductief leren van woorduit-

spraak. De resultaten van deze empirische studie staan ons toe om te stellen dat *inductieve leeralgoritmen in staat zijn om de uitspraak van woorden te leren met een bevredigend generaliseringsvermogen, zelfs wanneer de taakdefinitie (en ook de systeemarchitectuur) geen enkele van de abstractieniveaus reflecteert die expliciet als noodzakelijk worden verondersteld door taalkundige theorieën.*

Hoofdstuk 1 introduceert de bovengenoemde historische achtergrond van argumenten en tegenargumenten met betrekking tot de haalbaarheid van inductief leren van natuurlijke taal. Op basis van claims van de structurele taalkunde en het automatisch leren wordt vervolgens de probleemstelling geformuleerd.

In Hoofdstuk 2 wordt een overzicht gegeven van relevante achtergrondkennis. Er worden drie groepen inductieve leeralgoritmen beschreven die (in principe) geschikt zijn om woorduitspraak te leren: connectionistisch leren, instantie-gebaseerd leren, en het leren van beslissingsbomen. Het hoofdstuk biedt een overzicht van heersende taalkundige ideeën over morfologie en fonologie, en legt de nadruk op de abstractieniveaus die aanwezig worden verondersteld bij het uitspreken van woorden. Vervolgens wordt de gegevensbron beschreven waaruit de in de studie gebruikte voorbeelden van de uitspraak van woorden zijn gehaald: de Engelse lexicale data base van CELEX. Tenslotte wordt de methodologie beschreven die door de hele studie heen gevolgd is.

Hoofdstuk 3 beschrijft de toepassing van de geselecteerde leeralgoritmen op vijf deeltaken van de uitspraaktaak. De deeltaken representeren vijf taalkundig gemotiveerde abstractieniveaus binnen woorduitspraak: (i) morfologische segmentatie, (ii) grafemische ontleding, (iii) grafeem-foneem-omzetting, (iv) lettergreepsplitsing, en (v) klemtoontoekenning. De resultaten behaald met het toepassen van de vijf algoritmen op ieder van de deeltaken laten zien dat de algoritmen een redelijk tot excellent generaliseringsvermogen kunnen halen. Daarnaast laten de resultaten zien dat hoe minder een leeralgoritme abstraheert over het leermateriaal door gegevenscompressie, des te beter zijn generaliseringsvermogen is, op iedere deeltaak.

In Hoofdstuk 4 worden modulaire woorduitspraaksystemen geconstrueerd, geleerd en getest. De architectuur van de modulaire systemen is geïnspireerd op twee bestaande tekst-naar-spraak-systemen. Beide modulaire systemen voeren de uitspraaktaak uit in sequentieel verwerkende modules. De behaalde generaliseringsscores geven aan dat opeenstapelingen van fouten, doorgegeven van module naar module, de prestatie van de systemen ernstig hinderen. Om een gedeelte van deze ongewenste fouten

te ondervangen laten we het idee van een systeem met vijf modules varen en onderzoeken we twee systemen met drie modules. In deze drie-module systemen worden twee paren van deeltaken van de vijf-module systemen geïntegreerd tot enkele deeltaken. De precisie van deze systemen, die onderscheid maken tussen (i) morfologische segmentatie, (ii) grafeem-foneem-omzetting en (iii) klemtoontoekenning, is significant beter dan die van hun tegenhangers met vijf modules.

In Hoofdstuk 5 wordt het idee van sequentiële modulariteit vervangen door dat van parallelle modulariteit. Drie parallel-modulaire systemen worden getest. In het eerste systeem word woorduitspraak uitgevoerd door een enkele module, die letters omzet naar fonemen met klemtoonmarkeringen in een enkele omzettingsslag. Het tweede systeem voert twee deeltaken parallel uit, namelijk de omzetting van letters naar fonemen, en de omzetting van letters naar klemtoonmarkeringen. In het derde systeem wordt de letter-foneem-omzettingsdeeltaak verder uitgebreid tot 25 partiële deeltaken: ieder van deze partiële deeltaken representeert de herkenning van één articulatorisch kenmerk van het te classificeren foneem. De resultaten wijzen uit dat het systeem met de enkele module en het systeem met de twee modules beter presteren, en dat het articulatorisch-kenmerk-detectiesysteen slechter presteert dan het best presterende drie-modulesysteem beschreven in Hoofdstuk 4 dat getraind is met hetzelfde leeralgoritme.

Hoofdstuk 6 introduceert drie 'poortwachtersystemen' (*gating systems*) voor woorduitspraak waarin geen taalkundige kennis verwerkt is. De woorduitspraaktaak wordt in deze systemen opgesplitst in twee parallel-verwerkte partiële woorduitspraaktaken. In plaats van de taak op te splitsen op het uitvoerniveau wordt de taak opgesplitst op basis van het toepassen van een poortwachtercriterium op het invoerniveau: de spelling van (delen van) woorden. Drie poortwachtersystemen worden getest, te weten die met een toevalsgebaseerde poortwachter, een typicaliteits-gebaseerde poortwachter, en een voorkomen-gebaseerde poortwachter. Uit de resultaten blijkt dat in het toevalsgebaseerde poortwachtersysteem de woorduitspraaktaak slechter wordt geleerd dan de ongesplitste taak in zijn geheel. In de typicaliteits-gebaseerde en voorkomen-gebaseerde systemen wordt de taak met dezelfde precisie geleerd als de ongesplitste uitspraaktaak. Het aanbrengen van een poortwachter leidt niet tot verbetering van de prestatie, maar het is mogelijk om met een poortwachtersysteem de woorduitspraakgegevens automatisch op te delen in essentieel verschillende deelverzamelingen van gegevens.

Een samenvatting van de resultaten uit de Hoofdstukken 3 tot en met 6

wordt gegeven in Hoofdstuk 7. Speciale aandacht wordt besteed aan de mate van computationele efficiëntie van de gebruikte leeralgoritmen. Met instantie-gebaseerd leren worden de beste resultaten geboekt; de systemen die door het leren van beslissingsbomen worden gegenereerd bieden het beste evenwicht tussen generaliseringsvermogen en computationele efficiëntie. Hierop volgend worden analyses op de woorduitspraak-gegevens uitgevoerd om de oorzaken te zoeken voor het succes van instantie-gebaseerd leren: de analyses wijzen uit dat instanties van woorduitspraak in families voorkomen, die bestaan uit kleine aantallen leden met dezelfde klasse. Instantie-gebaseerd leren is in staat om goed te presteren op verzamelingen gegevens die dit soort kleine families (*small disjuncts*) bevatten. Vervolgens beschrijft het hoofdstuk een modulair woorduitspraak-systeem dat een aantal empirische bevindingen aangaande verschillende in het proefschrift onderzochte systemen combineert. Een analyse van het systeem toont aan dat het mogelijk is om betere woorduitspraaksystemen te bouwen door taalkundige en empirische kennis te combineren. Tenslotte biedt het hoofdstuk een overzicht van verwant onderzoek, en geeft een aantal indicaties voor verbetering van de huidige aanpak en onderwerpen van toekomstig onderzoek.

In Hoofdstuk 8 wordt de conclusie getrokken dat inductieve leermethoden, in het bijzonder instantie-gebaseerde leermethoden, in staat zijn om woorduitspraak te leren en daarbij een adequaat generaliseringsvermogen te bereiken. Taalkundige voorkennis in de taakdefinitie (en in de systeemarchitectuur) kan beperkt worden tot een absoluut minimum, kan met andere woorden *impliciet* gelaten worden, zonder dat het generaliseringsvermogen daar onder lijdt.

Curriculum Vitae

Antal van den Bosch werd op 5 mei 1969 te Made geboren. In 1987 behaalde hij het diploma Gymnasium B aan het Gertrudislyceum te Roosendaal. In datzelfde jaar begon hij een studie Letteren, in het bijzonder Taal en Informatica, aan de Katholieke Universiteit Brabant (KUB) te Tilburg. Deze studie werd in 1992 afgesloten met een doctoraal examen. Als erkend gewetensbezwaarde werkte hij vervolgens op het Instituut voor Taal- en Kennistechnologie (ITK) en bij de vakgroep Psychologie van de Faculteit der Sociale Wetenschappen, beide aan de KUB. Aansluitend was hij voor een periode van drie maanden verbonden aan het Laboratoire de Psychologie Expérimentale aan de Université Libre de Bruxelles.

In april 1994 trad hij als assistent in opleiding (AIO) in dienst van de Vakgroep Informatica van de Faculteit der Algemene Wetenschappen aan de Rijksuniversiteit Limburg (sinds september 1996 bekend als Universiteit Maastricht). Hij werd begeleid door prof. dr H. J. van den Herik, dr A. Weijters, en dr E. O. Postma. Per 1 september 1997 vervult hij de functie van post-doc onderzoeker, gefinancierd door NWO, binnen het project Induction of Linguistic Knowledge (ILK), dat onder leiding staat van dr W. Daelemans, en plaatsvindt aan de KUB in Tilburg.